NEW APPROACHES FOR REFACTORING TO FRAMEWORKS

BY

FAISAL MOHAMMED ABOBAKR BANAEAMAH

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

JUNE 2009

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **FAISAL MOHAMMED BANAEAMAH** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

Dr. Mohammad R. Alshayeb (Advisor)

Dr. Mahmoud O. Elish (Member)

Dr. Sabri A. Mahmoud (Member)

Dr. Kanaan A. Faisal
(Department Chairman)

Dr. Salam A. Zummo
(Dean of Graduate Studies)

8/2/12
Date

Dedication

To my father's sole, my mother and the rest of my beloved family.

Acknowledgements

I would like to take this chance to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for all support extended during this research.

Furthermore, I would like to express my gratitude and appreciation to my thesis advisor, Dr. Mohammad Alshayeb, for his continuous support, unlimited help, valuable guidance and advice since we first met. My gratitude is also due to the thesis committee members, Dr. Mahmoud Elish and Dr. Sabri Mahmoud for their help and enlightening comments.

Last, but not least, I would like to thank my mother, my brothers, my sisters, my nephews, and my nieces for their prayers, encouragement, and continuous support. Also, thanks to my friends for their suggestions and encouragement during the preparation of this thesis.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABSTRACT

Name: Faisal Mohammed Abobakr Banaeamah

Title: New Approaches for Refactoring to Frameworks

Major Field: Computer Science

Date of Degree: June 2009

Refactoring to framework is a software refactoring process which is applied on an existing software application to produce reusable domain classes for specific problem domains while improving their quality. These produced classes can be used in the development of other applications. There are several processes exist to help designers create application frameworks, however, none of these processes provide guidelines to refactor existing software applications to frameworks. In this thesis, we propose two approaches for refactoring to framework composing of a set of refactoring methods; the quality attribute based refactoring to framework (QARtF) and the level based refactoring to framework (LRtF). These two approaches provide a standard approach for application frameworks development. They are empirically applied on real software applications. This study, in turn, helps the designers to build application frameworks from software applications using the proposed approaches of refactoring to framework.

ملخص الرسالة

الاسم: فيصل محمد أبوبكر باناعمة

عنوان الرسالة: طرق جديدة لإعادة الهيكلية إلى أطر برمجية عامة

التخصص: علوم الحاسب الآلي

تاريخ التخرج: يونيو 2009

يمكن تعريف عملية إعادة الهيكلية إلى إطار برمجي عام على أنها عملية تنفيذ إعادة الهيكلية على برامج موجودة سلفاً لإنتاج مجموعة من العناصر البرمجية المتخصصة في مجال محدد من خلال تطوير صفات الجودة. العناصر البرمجية الناتجة عن هذه العملية يمكن استخدامها في تطوير برامج أخرى. على الرغم من وجود عدد من الإجراءات والعمليات التي تساعد مصممي ومطوري البرامج في صناعة الأطر البرمجية، إلا أنه لا يوجد منها مايعتمد على إعادة الهيلكية لإنشائها. في هذا البحث، نقترح طريقين لإعادة الهيكلية إلى أطر برمجية عامة يتكونان من مجموعة من أساليب إعادة الهيكلية؛ إعادة الهيكلية إلى إطار بناء على صفات الجودة، وإعادة الهيكلية إلى إطار بناء على مستوى العنصر البرمجي. هذان الطريقان يقدمان نموذجاً محدداً لبرمجة الأطر العامة. تم تطبيق هذه الطرق عملياً على برامج حقيقية. هذه الدراسة بدورها، تساعد مصممي البرامج في إنشاء أطر برمجية عامة من برامج موجودة سلفاً عن طريق استخدام الطرق المقترحة لإعادة الهيكلية.

CHAPTER 1

INTRODUCTION

High quality of a software application is a major point of its success. The software quality of the software application is determined by set of criteria; software design quality, mapping design to requirements and quality attributes which satisfy design and requirements [1]. A set of nine software measurements or attributes constitutes high quality of the software application in requirement level. The nine software measurements and attributes are correctness, unambiguousness, completeness, consistency, ranking of importance and stability, verifiability, modifiability, traceability and understandability [1]. There is a set of software attributes relates to the software design. Examples of the software design attributes are modularity, flexibility, extensibility [2], maintainability, reusability [2, 3] and understandability [3]. However, there are some proposed approaches and processes used to improve the software quality and satisfy quality needs.

Application frameworks are semi-complete applications that can be reused to produce custom applications [2]. The application frameworks tend to improve software quality through building reusable components. These reusable components can be used further by other applications. However, the application frameworks improve software quality by localizing the impact of design and implementation changes. Thus, the effort required for understanding and maintaining existing software is reduced [2].

Software refactoring is restructuring internal structures of an existing software application without affecting its external behaviors. The software refactoring can also be used to improve the software quality [3]. In other words, the internal quality attributes of the software application can be improved via the internal restructuring of the software refactoring.

Refactoring to framework is a software refactoring process which is applied on an existing software application to produce an application framework (domain classes) from the classes of the application. In refactoring to framework, the internal structure of the software application is changed to improve certain software quality goals. These software quality goals categorize the common application frameworks features and characteristics. The produced set of domain classes (framework) can be reused in future to implement other custom applications.

The main objective of this research is to propose a new approach for refactoring to framework. The refactoring to framework process suggests a sequence of refactoring methods to be applied on any software application in order to produce a set of domain class that can be used as a framework. The refactoring methods of the refactoring to framework process are empirically applied on two different software applications with different scales and domains.

The following sections describe the research problem statement, the rationale of the research problem, the contributions of this thesis and the organization of the thesis.

1.1. PROBLEM STATEMENT

Refactoring to framework is a software restructuring or refactoring process to produce a common application framework for a specific problem domain (reusable domain classes). It is used to produce domain classes of a software application to be more common and reusable for future custom applications. The produced application framework can be reused in development of other applications. There are a number of methods exist that help designers create frameworks [4, 5], however, none of these provide guidelines or steps to refactor existing software applications to frameworks. Therefore, we need to propose a process for the refactoring to framework which consists of a set of refactoring methods, or refactorings that can be applied in sequence to produce a framework. The process is empirically applied on existing applications.

1.2. RATIONALE: PROBLEM IMPORTANCE

The refactoring to framework approach produces an application framework (i.e. reusable domain classes). It is expected that the refactoring to framework improves the software quality of software applications. This is because the application frameworks enhance software quality via building reusable components [2]. Therefore, the software quality of the software applications is improved when reusing the application frameworks. Moreover, the refactoring to framework approach also provides a standard approach for application frameworks development, which are difficult to developed and do not have certain standards [2].

1.3. RESEARCH CONTRIBUTIONS

The objective of this research is to propose a new approach for refactoring to framework. The main contribution of this work is to define the refactoring to framework approach as follows:

1. Identify software quality goals or attributes of the application frameworks.
2. Specify the refactoring methods positively contribute in the quality goals of the application frameworks.
3. Define a refactoring to framework approach consisting of phases of the specified refactoring methods to produce a framework.
4. Empirically apply the defined refactoring to framework approach on existing software applications and calculate system-level software metrics.

1.4. THESIS ORGANIZATION

The rest of the thesis is organized as follows. Chapter 2 provides a background on application frameworks and software refactoring. Chapter 3 presents literature review and an overview of related works. Chapter 4 specifies and defines the external application frameworks quality attributes and their enhancement processes. Chapter 5 presents a set of selected refactoring methods positively affect the application frameworks quality. Chapter 6 proposes a refactoring to framework approach and empirically applies it on existing

applications. Chapter 7 concludes the thesis with summary of main contributions and directions for future work.

CHAPTER 2

BACKGROUND

In this chapter, we provide an overview of application frameworks and software refactoring. The chapter includes definitions, purposes and goals, activities or classifications and explanations of each topic.

2.1. APPLICATION FRAMEWORKS

The application framework can be defined as a semi-completed application that can be reused and specialized to produce custom applications [2]. It is also defined as is a set of cooperating classes with a reusable design for specific domain software applications [4]. However, the application framework is customizable to particular applications by subclassing and composing instances of their class [4].

2.1.1. IMPORTANCE OF APPLICATION FRAMEWORKS

Development of complex software applications from scratch is expensive and error-prone because they rely on continuous rediscovery and reinvention which are major source of development testing cost [2]. An existing reliable software application can be used to develop a new problem domain application [6]. If an existing high quality application is reliable and reusable, then it positively contributes to the quality of software

applications developed by reusing it [2]. Therefore, the application frameworks simplify software design and implementation. Hence, the application frameworks are used to reduce development cost and to improve software quality [2].

However, the primary characteristics of the application frameworks are modularity, reusability, extensibility and inversion of control [2]. They provide a reusable environment for software components and facilitate easy software components development [6]. They enhance the modularity of software by encapsulating functionalities behind stable interfaces in order to restrict and localize the changes on software design and implementation [2]. They also improve the reusability of software by defining generic software components [2]. The application frameworks support the extensibility of software by providing explicit stable interfaces for services and features to ensure customization [2].

2.1.2. APPLICATION FRAMEWORKS ARCHITECTURE

The software architecture of application frameworks is represented in terms of classes, methods and objects. The architectural design of application frameworks consists of abstract classes and their responsibilities and collaborations [4]. The application frameworks have abstract architectures. The abstract architecture relies on four aspects: structure, functionality, abstraction and reuse [4]. The structure is a composition of classes with specific responsibilities and collaborations [4]. The functionality provides the required domain features of software application [4]. The abstraction is represented as generalized structures and functionalities [4]. The reuse is expressed by communalities

among a group of software applications to be reused in applications development [4]. However, the main architecture and design concepts of the application framework can be summarized as inheritance [2, 4, 5], abstraction [2, 4-6], generalization [2, 4], encapsulation [2, 4, 5] and dynamic binding [2].

2.1.3. APPLICATION FRAMEWORK DEVELOPMENT

The application framework development mainly consists of three major steps: Domain Problem Expertise Solutions, Protocol Contents Analysis and Object-Oriented Application Framework Creation using CASE [5]. First, the application framework development starts with studying and solving sample representative domain problems by domain experts and solutions are provided [5]. Second, the application framework development uses content analysis methodology in combination with the use of patterns. The content analysis classifies the solutions of the domain problem into categories with similar meaning. In the analysis, protocols are generated by discovering content categories and defining relationships among the content categories [5]. Finally, the application frameworks should be modeled in a Computer Aided Software Engineering (CASE) environment capable of translating the model into different programming language. They can be modeled by the Unified Modeling Language (UML). The modeling process of the application frameworks aims to represent the solution protocols with more stable relationships. Thus, the protocols of the domain problem are mapped to classes, attributes, relationships, or behaviors [5].

2.1.4. APPLICATION FRAMEWORKS EXAMPLES

There are many application frameworks widely used by application developers. MacApp [2] is a primary object oriented application framework for the Mac OS. Moreover, InterViews is an object-oriented framework for graphical user interfaces [2]. Microsoft Foundation Classes (MFC) is a contemporary GUI application framework to develop graphical software applications [2]. In addition, Distributed Component Object Model (DCOM) [2] is a Microsoft application frameworks for communication of software applications distributed across multiple networks. Java Remote Method Invocation (Java RMI) [2] is an application framework providing object-oriented interface for remote procedure calls. Moreover, there are application frameworks implementations of Common Object Requesting Broker Architecture (CORBA) in multiple programming languages to support running on multiple computers in contemporary software development [2]. However, there are many other application frameworks such as ET++ [2], ACE [2], Microsoft .Net Framework, Oracle Application Framework, etc.

2.1.5. APPLICATION FRAMEWORKS CLASSIFICATION

The application frameworks can be classified based on either their scope or extension techniques. Based on scope, they are divided into system infrastructure, middleware integration and enterprise application frameworks [2]. The system infrastructure frameworks simplify portable and efficient system infrastructure development such as operating system [2]. The middleware integration frameworks are market framework products integrating distributed applications and components [2]. The

enterprise application frameworks implement enterprise business activities to provide substantial return on investment and support end-user product directly [2]. Based on extension techniques, they are categorized to white-box and black-box frameworks [2]. The white-box application frameworks rely on object-oriented features such as class inheritance and method overriding. The black-box application frameworks are extended by defining interfaces for pluggable software components via object composition [2].

2.1.6. APPLICATION FRAMEWORKS DEVELOPMENT LIMITATIONS

Although the application frameworks improve software development, they have some difficulties or limitations in implementation or development. They are complex to develop and need effort to reuse [2, 6]. They are difficult to integrate [2], maintain [2, 6] and adapt due to requirements changes [2]. Usually, they cost execution time and storage efficiency [2]. Finally, the lack of standards for designing and developing application frameworks is considered as a main limitation of the application frameworks [2, 6].

2.2. SOFTWARE REFACTORING

Software refactoring is “a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior” [3]. It can also be defined as “the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure” [3]. However, its definition can be generalized as “the art of improving the

design of existing code" [7]. It is code restructuring or reengineering without affecting its functionalities. It provides approaches to recognize problematic code and methods to improve design of the code after it has been written [3, 7]. Bad smells [8], or code smells [9] are structures of software code needed to be refactored [8]. They describe potential problems in code [9]. Refactoring methods, refactoring rules or refactorings [8] are set of principles, techniques and practices of the software refactoring. They provide solutions to restructure bad code smells to improve software quality [8].

2.2.1. IMPORTANCE OF SOFTWARE REFACTORING

The software refactoring mainly aims to improve software design of application in order to make it easier to understand [3] and eliminate code smells [9]. An important aspect of improving design is to eliminate duplicate code for future modification support [3]. The software refactoring is used to improve software quality such as extensibility, reusability, maintainability [8, 10], modularity, complexity and efficiency [10]. For example, when software is refactored for maintainability, it helps to make software easier to maintain.

Software applications are hard to modify and adapt when they are hard to read, contain duplicated logic, require additional behavior to change running code, or have complex conditional logic [3]. Moreover, modifications and adaptations of the software application may result more code complexity due to changes in requirements [10]. Hence, it may cause reducing software quality [10]. Therefore, software refactoring facilitates the modifications and the adaptations of the software application [3].

2.2.2. GUIDELINES FOR SOFTWARE REFACTORING

There are three rules or guidelines for the software refactoring. First, the software refactoring is applied when a new function is added in order to improve understandability [3]. Second, the software is refactored when a bug needs to be fixed to make code more understandable [3]. Third, the software refactoring is used when performing a code review in order to spread and pass knowledge among a development team [3].

2.2.3. SOFTWARE REFACTORING LIMITATIONS

Software refactoring has some limitations and difficulties on some software implementations, such as database applications [3]. The database applications are difficult to refactor because of tight coupling between business components and database schemas and data migration [3]. Moreover, software refactoring should not be applied in two situations. First, it should not be applied when it is easier to start development of the software application from the beginning than to refactor it [3]. Second, it should be avoided when development of software application is close to a deadline [3]. In addition, a common problem with the software refactoring may reduce the performance of software by causing the software to run more slowly while making the software more understandable [3].

2.2.4. SOFTWARE REFACTORING ACTIVITIES

Software refactoring mainly involves a set of two processes. First, source of problems should be recognized in the software source code [7]. Second, the refactoring techniques should be applied on the software [7]. Implicitly, the software refactoring encompasses four tasks: finding where to apply a change, choosing how to perform the modification using refactoring methods, guaranteeing not altering the external behavior or functionality of the software and ensuring improving in the internal structure of the software [9, 10]. The refactoring should also maintain the consistency between the refactored code and other software artifacts and documentations [10].

CHAPTER 3

LITERATURE REVIEW

Software refactoring has been studied extensively. These studies include researches in improving quality using refactoring, refactoring processes, and automating refactoring using tools. This chapter provides an overview of the previous studies. It summarizes previous studies that investigated the refactoring processes and quality improvement, architecture-level refactoring, experience-based and cascade refactoring, refactoring tools, and scheduling refactoring conflict.

3.1. QUALITY IMPROVEMENT USING REFACTORING

Tiarks discussed refactoring a system due to specific quality requirements, such as the maintainability [11]. He considered relationships and dependencies between quality goals of software. The analysis included positive or negative effects of refactoring due to certain software quality goals such as maintainability on other quality goals such as performance. The result was a framework to visualize quality requirements and their dependencies using software metrics to evaluate and measure the refactoring quality [11].

Tahvildari showed a framework of re-engineering of object-oriented systems to improve software quality [12]. The framework considered specific design and quality requirements namely, performance and maintainability. In the framework, the quality requirements, or goals, were represented using Soft-goal Interdependency Graphs (SIG).

The software transformations, or refactorings, were accomplished as a sequence of iterative and incremental steps to the source code. The evaluation procedure upon each transformation step was done using software metrics.

3.2. REFACTORING PROCESSES

There are many refactoring processes which are followed to perform refactoring on applications. Some processes target implementation phase while others target the design phase. In this section, we present a review of some of these processes.

A common refactoring process can be accomplished by certain steps. First, define Soft-goal Interdependency Graphs (SIG) to quantify each soft-goal, which is a non-functional requirement or a quality goal to be satisfied and its contributions or dependencies to other soft-goals in metric measurements. Next, prioritize and order soft-goals due to the software metrics measured of each soft-goal. Then, apply effective refactoring transformations per each soft-goal and evaluate using software metrics. This process is done in sequence and iteratively [11, 12].

Tahvildari et al. proposed a refactoring process for object-oriented legacy systems to improve the fulfillment of the non-functional requirements, such as reusability, performance and maintainability. A legacy system is an old application program that continues to be used because the user does not want to replace or redesign it [13]. In this process, design patterns and refactoring operations are used in order to enhance software

quality goals during re-engineering. They used a remediation specification component to select optimal transformations or refactoring operations.

Geppert et al. applied refactoring on a part of a large legacy business communication product [14]. They proposed a number of strategies and effects of the refactoring effort for changeability.

Any software component consists of a specification and a realization [15]. While the specification consists of a structural, a behavioral and a functional model, the realization consists of a refined structural model, an activity model and an interaction model, as shown in Figure 3-1 [15]. Software refactoring can be applied to several models of specification and realization of software components [15]. Kolb et al. described several-model refactoring processes of systematic refactoring of an existing software component for reuse in a product line by improving the design and implementation for reusability and maintainability [15]. A software product line refers to a collection of similar software systems from a shared set of software assets using a common means of production [15]. They proposed a three-step approach that provides an overview of activities done while re-documenting and redesigning the component as follows:

- Perform automatic improvements of the whole code,
- Improve selected aspects of the complete code manually and finally, and
- Improve the design and implementation of selected sub-components [15].

As advantage of their work, maintainability, reusability and hence suitability of a legacy product line are increased [15]. However, their work is limited to quality needed for legacy system, not for any type of application.

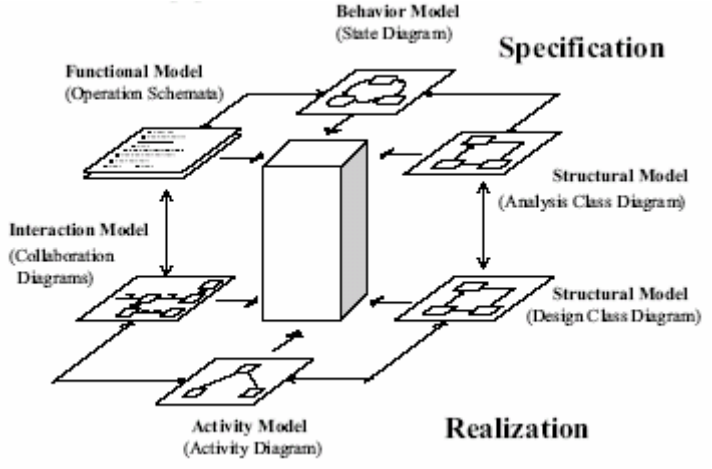

Figure 3-1: Refactoring Models [15]

Lee et al. applied an object-oriented refactoring process on a project called D2DTM, which is a case tool for developing windows system [16]. They focused on system reusability, performance and cost reduction by optimizing structures and classes. On the other hand, Xing and Eleni examined refactoring with evolution history on many client applications to come up with some requirements and design suggestions for tools to support refactoring [17].

Refactoring with contract [18], or refactoring by contract (RbC) [19], is another process of refactoring. It is a refactoring technique to verify refactoring based on contracts [18, 19]. The contract is composed of preconditions, post-conditions and invariants [18, 19]. The precondition specifies the conditions of refactoring [19]. The post-condition

states conditions to be verified after refactoring [19]. The invariant identifies conditions to be preserved by refactoring [19]. The refactoring with contract makes less error-prone software development [18] and assists in building reliable and safe software evolution [19]. Goldstein et al. developed a tool called Crepe, which does refactoring with contract [18]. However, refactoring with contract must take some influences. Refactorings must be applied on the assertions of the contract as well as the code [18]. In addition, methodological implications of the contract have to be checked before performing refactorings [18]. Ubayashi et al. presented the refactoring by contract and contract writing language (COW) [19]. The contract writing language is a language for contract description based on first-order logic. It provides a set of primitive predicates to represent structural and behavioral program [19].

Bryton and Abreu proposed modularity-oriented refactoring process (MORe) and developed a MORe Eclipse plug-in tool [9]. The modularity-oriented refactoring is a cross-paradigm and language independent refactoring process, based on modularity metrics [9]. It mainly includes three certain steps. First, modularity metrics and code smells are collected and identified in the source code [9]. Then, a set of refactorings at the code smells are applied based on the modularity metrics [9]. Finally, the modularity improvements are identified by comparisons between the modularity assessment of the source code and the refactored code [9].

S. Shrivastava and V. Shrivastava presented, as a case study, a metrics based refactoring process to improve software quality [20]. First, source code of a software application is first assessed using software metrics. Then, the source code is refactored.

Finally, the refactored code is assessed using software metrics to measure the quality improvement. However, this refactoring process encompasses three phases: detecting code smells, selecting and performing refactorings, assessing quality improvement [20].

Extract Method is recognized as an important refactoring because it can be used with other refactorings to fix some design problems [21]. Tsantalis and Chatzigeorgiou proposed a methodology to automatically identify Extract Method refactoring opportunities [21]. The refactoring methodology adheres on preserving behavior of a program after refactoring, containing all computations of variables declarations in extracted code and duplicating extracted code in original method [21].

3.3. REFACTORING IN ARCHITECTURE LEVEL

Requirements changes increase the overall system complexity [1, 22]. Hence, software artifacts need to be restructured over their lifecycle in order to improve quality and decrease complexity. Refactoring of software architecture is the first step in maintaining system quality during evolution [22]. This is because software architecture is the highest level of design and implementation phases and the first step of matching requirements.

Ivkovic and Kontogiannis introduced a framework of software architecture refactoring using model transformations and quality improvement semantic annotations [22]. In this framework, the conceptual view of architecture is first represented using

UML 2.0 with corresponding stereotypes. Then, architecture models are annotated using soft-goals, metrics and constraints. Finally, their most suitable refactoring methods, such as extracting classes and methods, or moving attributes are applied.

Grunske and Neumann proposed a refactoring process focused on architecture specifications such as safety, reliability, maintainability, availability and temporal correctness [23]. Each component of application refactored should be annotated with an evaluation model such as Component Fault Trees (CFT). After that, the component that is needed to be refactored should be sent to a work unit called "Quality Improvement by Architecture Refactoring". Moreover, they introduced two transformation operators, which are Multi Channel Redundancy with Voting and Two Channel Redundancy [23].

3.4. EXPERIENCE-BASED REFACTORING

In agile processes, refactoring is important because of continuous improvement of software quality [24]. Unfortunately, applications of refactoring are subjective and heavily demand on the developers and result in an unstable quality assurance. Rech and Ras presented an experience-based approach for the semi-automatic and goal-oriented refactoring of software systems [24]. It is based on didactical augmented experiences, following the experience factory paradigm. This approach promises the accelerated acquisition, usability, reusability and learning of knowledge in the refactoring process. On the other hand, Dig and Johnson proposed another style of experience-based refactoring process to support detecting and classifying structural evolution for refactoring methods

using clone detection [25]. The incremental and low invasive integration of knowledge with e-learning approaches support refactoring by monitoring and planning of larger refactoring activities. However, it is limited to agile and rapid processes [25].

3.5. CASCADED REFACTORING PROCESS

Butler and Xu described a cascaded refactoring framework, as shown in Figure 3-2. The framework is designed by a set of models: feature model, use case model, architectural model, detailed design model and source code [26]. They presented a hybrid methodology of mixing the top-down modeling approaches and the bottom-up iterative refactoring approaches.

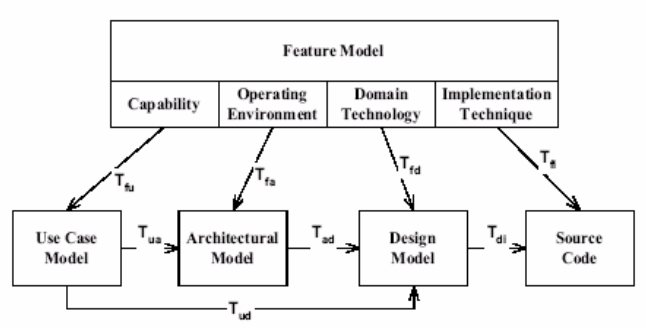

Figure 3-2: Cascaded Refactoring [26]

3.6. REFACTORING TOOLS

Refactoring tools automate refactorings rather than refactoring manually with an editor [27]. Many development environments of different programming languages include

refactoring tool [27, 28], for example, Eclipse, Microsoft Visual Studio, Xcode and Squeak [27]. The main advantages of the refactoring tools are to make the refactoring process less mistakable and faster [27], (i.e. quick and correct [29]). Murphy-Hill and Black proposed five principles for the refactoring tools [27]. First, the refactoring tool should allow programmer to choose desired refactorings quickly. Second, it is supposed to enable the user to switch faultlessly between program editing and refactoring. Third, the refactoring tool ought to allow view program code while using the tool. Fourth, it is supposed to avoid explicit configuration information. Fifth, the refactoring tool should support access all other tools of the development environment normally while using the tool [27].

Every refactoring tool requires a programmer to select a piece of code to be refactored and to interpret the refactoring precondition violations [29]. Murphy-Hill and Black induced a set of recommendations based on code selection and refactoring preconditions violation [29]. Based on code selection, the refactoring tool should be lightweight, help in code formatting and task specific [29]. Based on precondition violation, the refactoring tool should quickly comprehend precondition violation, indicate locations of a precondition violation, show precondition violations at once, easily distinguish precondition violation from warnings and advisories, indicate amount of work required to fix precondition violations, relationally display precondition violations when appropriate and provide distinguishable representations of precondition violations [29].

There are many refactoring tools available for a variety of programming languages [27, 28]. Most of refactoring tools are specialized for a particular language. Hence, they

differ from language to another based on syntax and semantics of the programming language even though refactoring operations are similar [28]. A definition of a process to extract code information for a suite of languages and appropriate refactoring operations increases the reusability and interoperability of refactoring tools. Marticorena et al. showed an implementation of a refactoring tool based on reuse engine among several languages [28].

Most of refactoring tools represent source code as an abstract syntax tree (AST). There is no standard way of representing and manipulating code. Hence, it is difficult to reuse, extend or customize. Mendonca et al. presented an XML-based refactoring framework, called RefaX [30], as shown in Figure 3-3. It provides standards to facilitate the development, customization and reuse of refactoring tools. The major advantage of RefaX is possibility and feasibility to develop refactoring tools independently of source code model, programming language and XML processing technology. Maruyama presented Jrbx as a refactoring tool that uses RefaX [31]. Jrbx uses control flow graphs (CFGs) and program dependence graphs (PDGs) for both pre-condition checking and change creation. The use of the XML, CFG and PDG representations makes the implementation of Jrbx more understandable and reusable.

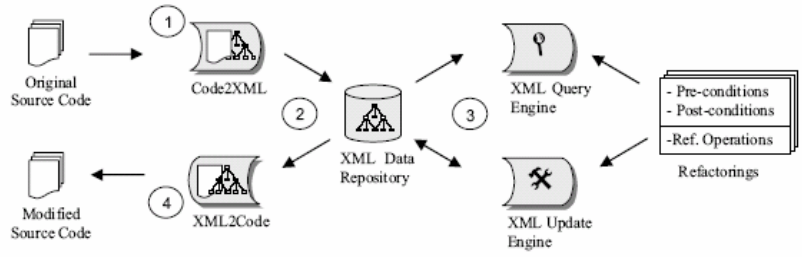

Figure 3-3: Refax Architecture [30]

3.7. REFACTORINGS CONFLICT SCHEDULING

Refactoring conflicts may occur between the software refactorings because a refactoring may delete or change necessary elements for other refactorings [8]. However, there are asymmetric and symmetric the refactoring conflicts. The asymmetric refactoring conflict between two refactorings occurs when the first refactoring can be applied only before the other refactoring [8]. On the other hand, the symmetric refactoring conflict between two refactorings occurs when they both cannot be applied together in any order [8].

Liu et al. suggested a conflict-aware scheduling approach via a heuristic algorithm [8]. It schedules refactorings based on a conflict matrix, which represents the refactorings and their effects on each other [8]. They concluded with that the refactoring with scheduling leads to great quality improvement than refactoring without explicit scheduling [8].

CHAPTER 4

APPLICATION FRAMEWORKS QUALITY ATTRIBUTES

The application frameworks can be identified by a set of primary features and characteristics. We named these framework features and characteristic as application framework quality attributes. The application framework quality attributes are set of external quality attributes which characterize application frameworks. The application frameworks quality attributes are reusability, modularity, extensibility, flexibility, maintainability [2] and usability [4]. Each of the application frameworks quality attributes is explained in the following sections.

4.1. REUSABILITY

Reusability is the ability of using software components or an existing source code in the development of another software application. It improves and increases software quality and productivity of the software application [32]. The reusability of software is one of the most essential benefits and characteristics of the application frameworks [2]. However, the reusability is affected by some other quality attributes; complexity, testability and modularity [32, 33].

4.1.1. REUSABILITY ENHANCEMENT PROCESSES

The reusability of software can be developed and improved by five procedures: extracting components, standardizing components, classifying components, retrieving components and adapting components [32]. The reusability of software is improved by extracting components from existing software systems [32]. Standardization of software components improves the reusability of software using a certain format of packaging [32]. Classification of software components based on attributes advances the reusability of software [32]. Retrieving software components from a software component library according to the software application attributes develops the reusability of software [32]. Selecting a suitable software component and adapting it to a new application also enhances the reusability of software [32]. Moreover, the reusability of software can be addressed by the generalization of software via generalizing common features of software [34]. The reusable software should extend and reuse existing software components [34, 35] and capture components constraints and design decisions [34]. The reusability of software also relies on abstract software architecture [35]. The reusability of software is enhanced by creating documentation and comments [34].

4.2. MODULARITY

Modularity is the ability of making software more consisting of separate independence parts or components, called modules [32]. The modularity of software is a primary feature of the application frameworks [2]. Modularity is affected by cohesion and

coupling [32]. Coupling is a measure of the degree of interdependence between modules and cohesion is an attribute of individual modules, describing their functional strength.

4.3. EXTENSIBILITY

Extensibility is the ability to extend a system, with minimum level of implementations and minimum impact to existing functions. It is a system design principle that takes into consideration future growth [36]. The extensibility of software is one of the primary benefits and features of the application frameworks [2].

4.3.1. EXTENSIBILITY ENHANCEMENT PROCESSES

The extensibility of software can be addressed and developed by some processes. The extensible software compromises interoperability by implementing non-functional features. The extensibility of software can be improved by preserving the compatibility of client and server (producer and consumer) model [36]. The extensible source code of software components should have proper identification to avoid confusion among them [36]. Moreover, the extensibility of software is enhanced by providing syntactic definitions for all extensible software components. It is as well improved by providing precise semantic definitions and documentations [36].

4.4. MAINTAINABILITY

Maintenance is the process of implementing corrective, adaptive or perfective software changes [37]. Maintainability is defined as the ability of a software application to be indicative of amount of effort necessary to perform maintenance changes [37]. In other words, it is the ease with which a software application or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [38]. The maintainability of software is a quality attribute of the application framework development [2]. The software maintainability can be estimated by the average maintenance effort and the complexity of software [37, 38].

4.4.1. MAINTAINABILITY ENHANCEMENT PROCESSES

The maintainability of software can be improved using hierarchical multidimensional design methodology. The hierarchical multidimensional design methodology consists of decomposing a software application into hierarchical levels from different dimensions; control, information and typography structures [37]. The control structure decomposes the software application into algorithms [37]. On the other hand, the information structure decomposes the software application based on data structure and flow [37]. However, the typography structure includes the typographic layout, naming and commenting of code [37].

4.5. USABILITY

Usability can be defined as the ease of use of a software application or product [39, 40]. It also means the level of ease to understand, learn, and use a software application [40]. One of the main features of the application frameworks is the usability of the framework [4]. The usability of software can be measured by different criteria such as the understandability of the software application [41]. The usability of software is affected by the functionality, consistency and self-explanatory of software components and messages [39].

4.5.1. USABILITY ENHANCEMENT PROCESSES

The usability of software can be enhanced and improved by several development processes or scenarios. The source code of software becomes usable through aggregating and combining data, long-running statements and multi-step commands [34]. The usability of software is improved by allowing cancel commands and operations. The usable software enables running multiple functions concurrently with no conflict [34]. Moreover, it should provide data validation, correctness and recovery from failure [34]. The usable software application maintains device independence and observes system state. It supports comprehensive searching in data and undo operation [34].

4.6. FLEXIBILITY

Flexibility is the ease with which a software application or component can be modified for use in applications or environments other than those for which it was specifically designed [42]. The flexibility of software is a feature characterizing the application frameworks [2]. The flexibility of software is affected by the evolution cost and evolution complexity [42].

4.7. QUALITY ATTRIBUTES MEASUREMENTS

Software metrics are categorized into three types: product metrics, process metrics and project metrics [43]. The product metrics identify internal characteristics or attributes of a product such as size, complexity, design, performance, etc [43]. The process metrics are used to improve software development and maintenance and include effectiveness of removing defects, testing pattern of defects and response time of fix process [43]. The project metrics represent project and execution characteristics and comprise number of developers over the software development life cycle of the software, cost, schedule and productivity [43].

Quality is composite of many characteristics thus quality is captured in a model that depicts the composite characteristics and their relationships [44]. A number of quality models were proposed: McCall et al. [45] proposed a quality model which was used as the basis for the first international standard for software quality measurement (ISO 9126).

Boehm et al. [46] proposed definitions and measures for a range of quality attributes, both McCall and Boehm described quality using a decompositional approach. IEEE also defined its own quality model [47]. These quality models define quality attributes (external attributes) and the metrics (internal attributes) used to measure these attributes. Since software metrics are measurable, each external attribute which is a function of a set of internal attributes can be easily measured and assessed. Briand and Wüst [48] have performed a comprehensive analysis of the empirical studies that exist concerning software quality models for Object-Oriented systems.

CHAPTER 5

REFACTORINGS IMPROVEMENTS ON QUALITY

In the previous chapter, we identified the application framework quality attributes. In this chapter, we compose a list of refactoring methods that positively contribute to the quality attributes of application framework. Then, we provide explanation of each refactoring method and its positive effects on the application framework quality attributes. Finally, we summarize the refactorings improvements on the application framework quality attributes.

5.1. SELECTED REFACTORINGS LIST

In this section, we select a list of refactorings consisting of 23 refactoring methods which contribute positively in quality attributes of the application framework. We rely on previous studies to determine positive effects of the refactoring methods on the quality attributes. However, some of these refactorings may affect the performance of the application which is not a core design attribute of framework as the framework causes performance degradation due to the additional overhead of dynamic invoking of methods [2]. The refactoring list covers all the categories of the refactorings in different structural levels: package, class, method, field, if-clause and iterative loop. These refactoring methods are among the most popular and most widely used methods.

5.1.1. ADD PARAMETER

Add Parameter is a very common refactoring that is used when a method needs more information from its caller which can be passed by an object [3, 50]. When a method requires information that was not passed in before, then it needs to be changed by adding a parameter [3]. Figure 5-4 illustrates and example. The necessary parameter is added to the method signature so that the method body uses the parameter's reference passed. However, Adding parameter reduces application-specific details and makes the application more flexible and extensible [2]. Therefore, the refactoring Add Parameter improves the *flexibility* and *extensibility* of the application framework.

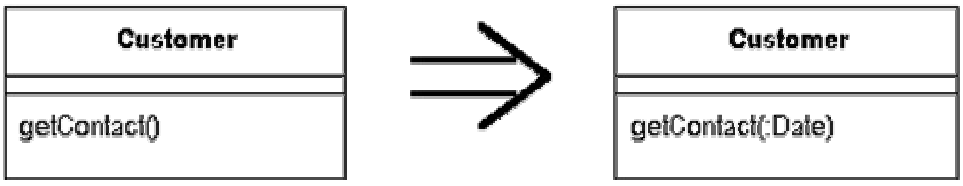

Figure 5-4: Add Parameter

5.1.2. DECOMPOSE CONDITIONAL

It is an extraction of methods from complicated conditions statements [3, 50]. Complex conditional logic is one of common complexity areas in software [3]. This refactoring decomposes the complex conditional logic into a method and replaces its chunks of code with the method call [3]. Decompose Conditional is shown in Figure 5-5. The complex condition is extracted into a separate method so that it will be replaced by a call of the extracted method. However, decomposing complex condition is an example of

decomposing complex algorithms as it is easy to maintain [37] as well as the decomposed method can be reused [21]. Therefore, the refactoring "Decompose Conditional" improves the *reusability* and *maintainability* of the application frameworks.

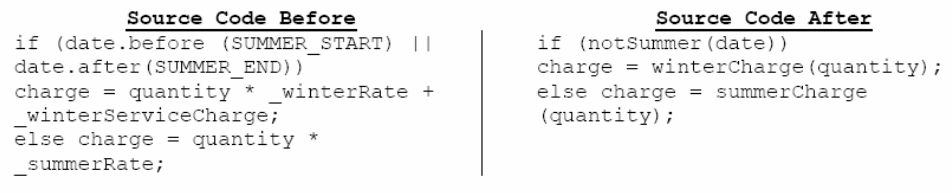

Figure 5-5: Decompose Conditional

5.1.3. ENCAPSULATE FIELD

When there is a public field in a class, it is changed to private and accessors are provided [3, 50]. Figure 5-6 shows a simple example of Encapsulate Field. The getting and setting methods are created for the field. This provides object encapsulation and data hiding by preventing public data [3]. It keeps the data and its behavior clustered together such that it makes the code easy to maintain, more modular [3] and so more reusable [32]. Therefore, the refactoring "Encapsulate Filed" improves the *reusability*, *modularity* and *maintainability* of the application framework.

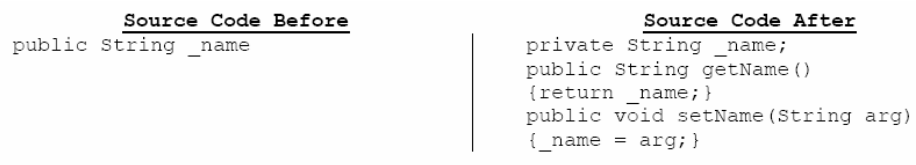

Figure 5-6: Encapsulate Field

5.1.4. EXTRACT METHOD

It can be implemented by grouping a code fragment together into a method with name that explains its purpose [3, 50]. Figure 5-7 shows an example of Extract Method. The particular code in the source method is extracted into a new method and is replaced with a call of the extracted method. Method extraction positively affects maintenance [21] because it simplifies the code by decomposing large methods into simple ones [21, 37]. It also creates new methods which can be reused [21]. Thus, Extract Method improves the *reusability* and *maintainability* of the application framework.

Source Code Before

```
void printOwing() {
printBanner();

//print details
System.out.println ("name:" +
_name);
System.out.println ("amount" +
getOutstanding());
}
```

Source Code After

```
void printOwing() {
printBanner();
printDetails(getOutstanding());}

void printDetails (double
outstanding) {
System.out.println ("name:" +
_name);
System.out.println ("amount" +
outstanding);}
```

Figure 5-7: Extract Method

5.1.5. EXTRACT PACKAGE

A package with too many classes and not easily understandable can be extracted into sub-packages based on dependencies or usages [50]. Sample Extract Package is presented in Figure 5-8. The extracted package groups relevant classes together. Class packaging makes the code more flexible because it makes classes' dependencies more explicit [50]. It only concentrates on packaging classes together, therefore, it does not

change the internal structures of classes, methods and fields because it. Thus, the refactoring "Extract Package" improves the *flexibility* of the application framework.

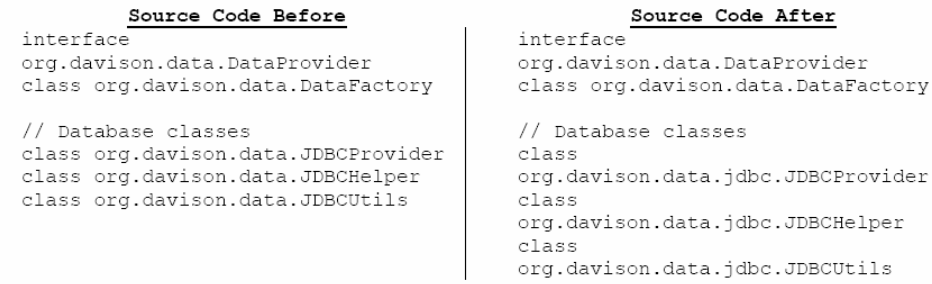

Figure 5-8: Extract Package

5.1.6. EXTRACT SUBCLASS

When a certain class contains a subset of features or members only used in some instances, a subclass is created for that subset [3, 50]. Extract Subclass example is shown in Figure 5-9. The extracted subclass is defined to inherit the source class. It has a particular behavior that differs from the behavior of its super-class. Subclass extraction is an implementation of inheritance and abstraction which are essential architecture designs of frameworks [2, 4-6]. The class abstraction provides more reusability [35]. Subclassing also reduces application-specific details in the source class and makes it more flexible and extensible [2]. Therefore, Extract Subclass improves the *reusability*, *flexibility* and *extensibility* of the application framework.

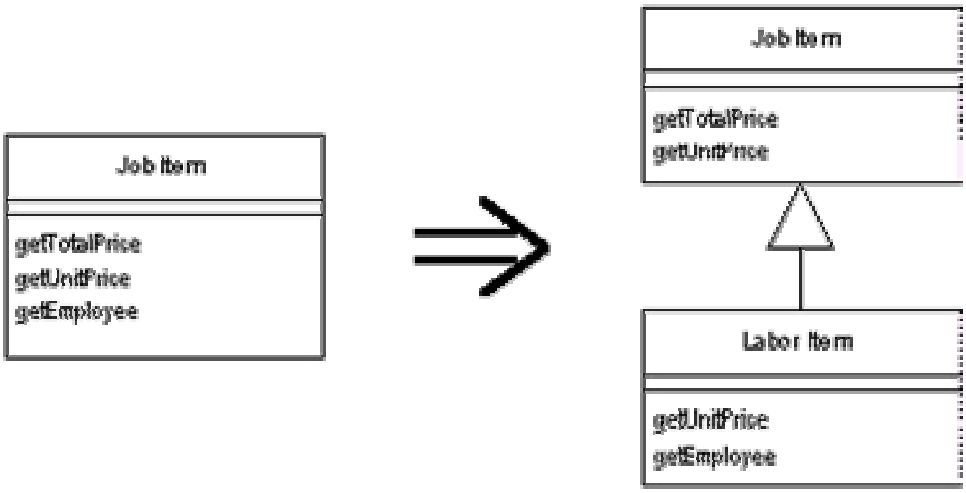

Figure 5-9: Extract Subclass

5.1.7. EXTRACT SUPER-CLASS

It is the extraction of a super-class from a set of classes with similar features or members to contain the common features and members [3, 50]. An example is shown in Figure 5-10. The extracted subclass is defined to be inherited by the source class. It has more general behavior than the behavior of its subclass. Super-class extraction is an implementation of inheritance, generalization and abstraction architecture design of frameworks [2, 4-6], which provides more reusability [35]. It includes also subclasses creation which provides more flexible and extensible application because it reduces application-specific details in the super-class [2]. Thus, Extract Super-Class improves the *reusability*, *flexibility* and *extensibility* of the application framework.

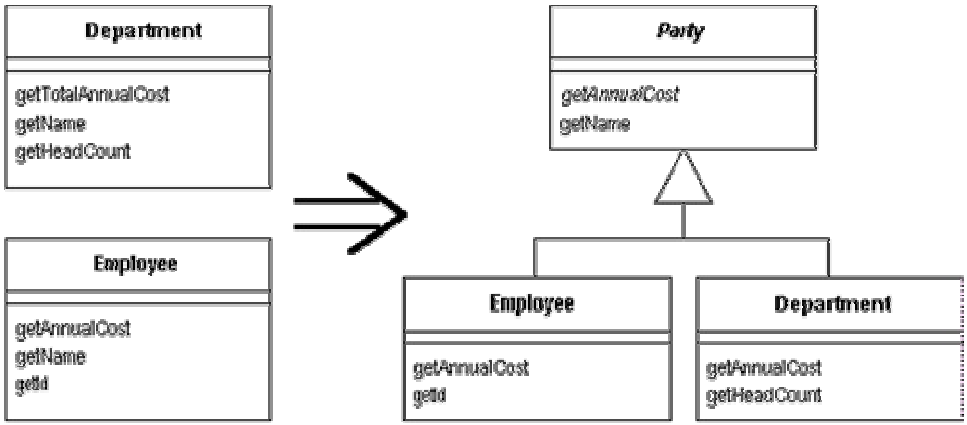

Figure 5-10: Extract Super-Class

5.1.8. HIDE DELEGATE

When a client calls a delegate class of an object, methods are created on the server to hide the delegate [3, 50]. An example of Hide Delegate is shown in Figure 5-11. A simple delegate method is created on the server for each method on the delegate. This refactoring provides objects encapsulation [3]. The modularity [2] and reusability [51] of the application are enhanced by encapsulation. It also facilitates maintenance because it limits the changes to the server and doesn't require propagation to the client [3]. Thus, the refactoring "Hide Delegate" improves the *reusability*, *modularity* and *maintainability* of the application framework.

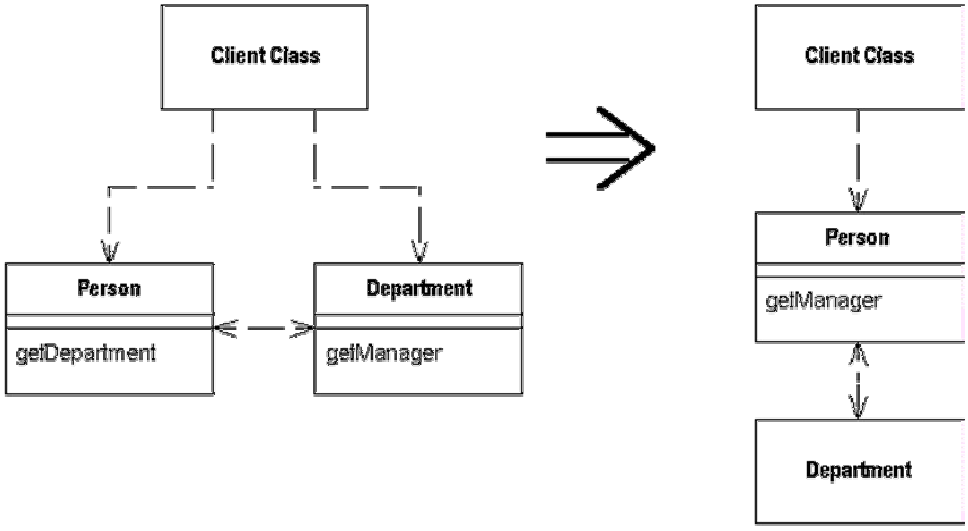

Figure 5-11: Hide Delegate

5.1.9. INLINE CLASS

When a class is not doing very much, all its features are moved into another class and that class is deleted [3, 50]. An example is illustrated in Figure 5-12. It assigns more responsibilities to the class. The source class is folded into another class by moving its members (methods and fields) [3]. The target class is more reusable as it contains more methods which can be reused [21]. Consequently, the refactoring "Inline Class" improves the *reusability* of the application framework.

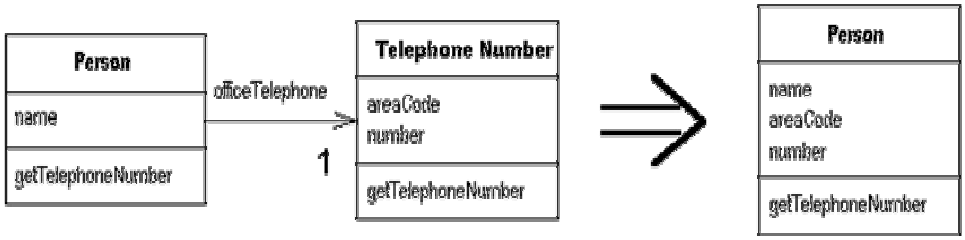

Figure 5-12: Inline Class

5.1.10. MOVE FIELD

If a field in a class is used by another class more than its class, a new field is created in the other class and all its users are changed to the new field [3, 50]. Move Field example is shown in Figure 5-13. The field is created in the target class and all of its references are replaced accordingly. This refactoring is considered when more methods on another class using the field than the class itself [3]. When a field is located in a class which uses the field much more than other classes, the cohesion of the classes is increased and the coupling among the classes is decreased [32, 33]. Therefore, Move Field improves the *reusability* and *modularity* [32].

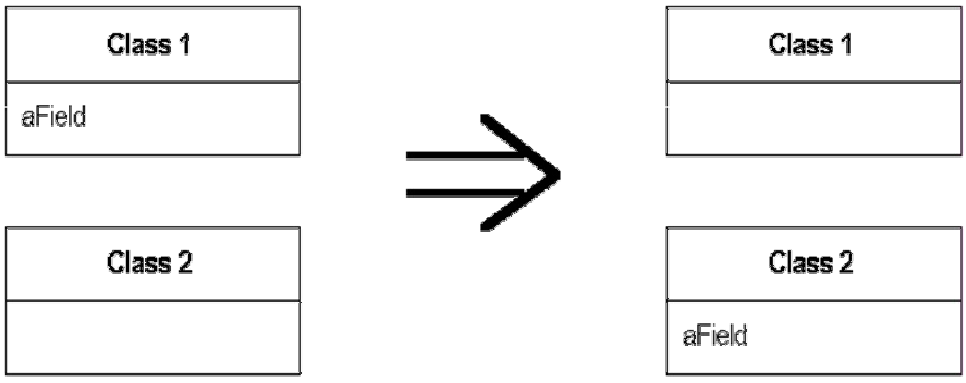

Figure 5-13: Move Field

5.1.11. PARAMETERIZE METHOD

When there are several methods do similar things but with different values contained in the method body, a new method is created using a parameter for the different values [3, 50]. The refactoring "Parameterize Method" is shown in Figure 5-14. The new method is created to have a signature with the common parameter so that it can be invoked by passing different values. Parameterization provides more flexible [2, 3] and extensible [2] application as it reduces application-specific details [2] and removes duplicate code [3]. Thus, the refactoring "Parameterize Method" improves the *flexibility* and *extensibility* of the application framework.

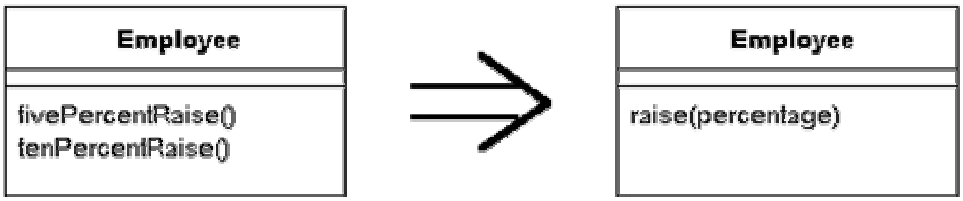

Figure 5-14: Parameterize Method

5.1.12. PULL UP FIELD

When two or more subclasses have same fields, the fields are moved to the super-class [3, 50]. Figure 5-15 shows an example of Pull Up Field. The common field among subclasses is pulled up to their super-class. This refactoring generalizes common fields by removing the duplicated data declaration and moving behavior of fields from subclasses to super-class [3]. It develops the inheritance and abstraction, essential design features of

frameworks [2, 4-6] and hence provides more reusability [35]. Thus, the refactoring "Pull Up Field" can be used to improve the *reusability* of the application framework.

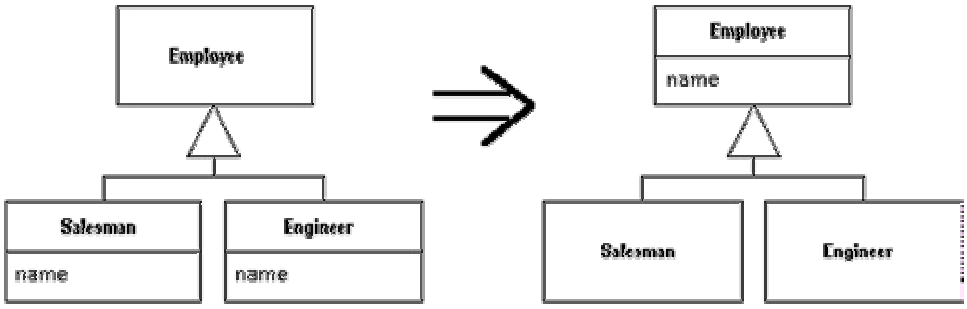

Figure 5-15: Pull Up Field

5.1.13. PULL UP METHOD

When a set of subclasses have methods with identical results, the methods are moved to the super-class [3, 50]. Pull Up Method is illustrated in Figure 5-16. The common method among subclasses is pulled up to their super-class. This refactoring reduces method duplication among classes and effort for alteration and maintenance by doing generalization and abstraction [3]. It develops generalization, inheritance and abstraction, which are essential architecture designs of frameworks [2, 4-6] and then provides more reusability [35]. Therefore, Pull Up Method can be used to improve the *reusability* and *maintainability* of the application framework.

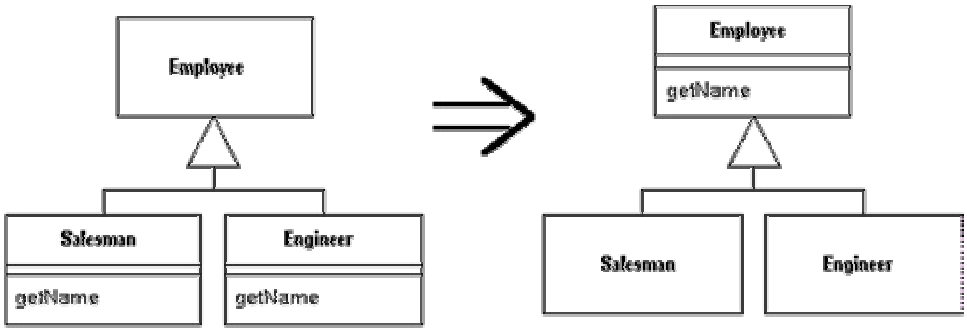

Figure 5-16: Pull Up Method

5.1.14. PUSH DOWN FIELD

When a field in a super-class is used only by some subclasses, the field is moved to those subclasses [3, 50]. Push Down Field is illustrated in Figure 5-17. The specific field in super-class is pushed down to the appropriate subclass. This refactoring moves behavior of fields to subclasses from super-class [3]. It improves the inheritance and abstraction, fundamental design features of frameworks [2, 4-6] and provide more reusability [35]. Thus, Push Down Field can be used to improve the *reusability* of the application framework.

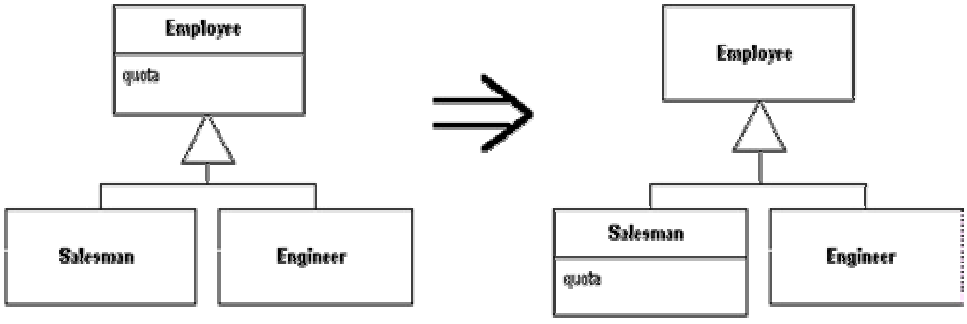

Figure 5-17: Push Down Field

5.1.15. REMOVE ASSIGNMENTS TO PARAMETERS

A temporary variable is used instead of assignment to a parameter [3, 50]. The refactoring "Remove Assignments to Parameters" is shown in Figure 5-18. The parameter should not be changed to refer to another object entirely. This avoids lack of clarity and confusion between pass by value and pass by reference [3]. It limits parameters to only represent the passed data to the method [3]. It also reduces confusion and provides consistency within the method body code [3], to be more maintainable [37]. Therefore, Remove Assignments to Parameters improves the *maintainability* of the application framework.

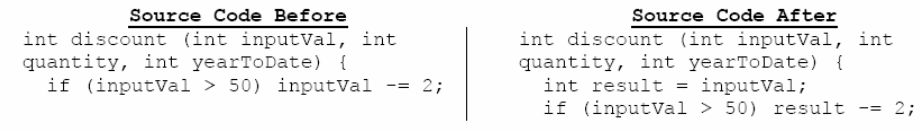

Figure 5-18: Remove Assignments to Parameters

5.1.16. REMOVE PARAMETER

A parameter, not used by method body, is removed [3, 50]. The refactoring Remove Parameter is illustrated in Figure 5-19. The unnecessary parameter is removed from the method signature so that the method can be called without passing the removed parameter. This refactoring reduces the number of parameters to be passed in method

calling and hence the method becomes easy to reuse [3]. Thus, Remove Parameter improves the *reusability* of the application framework.

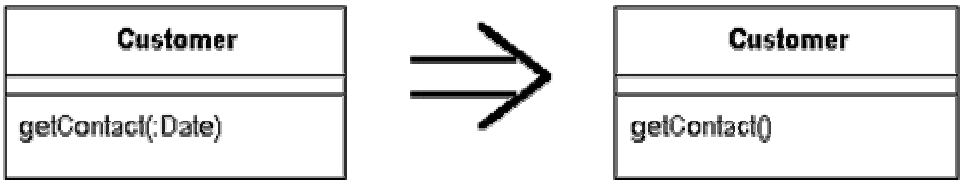

Figure 5-19: Remove Parameter

5.1.17. REMOVE SETTING METHOD

If a field is set at creation time and never altered, any setting method of that field is removed [3, 50]. Remove Setting Method is shown in Figure 5-20. The unneeded setter method is eliminated from the class. Providing a setter for a field indicates that a field may be changed after the object creation and initiation [3]. This refactoring keeps the necessary setting methods [3], which have high reuse potential, and hence provides more reusability [32]. Thus, Remove Setting Method improves the *reusability* of the application framework.

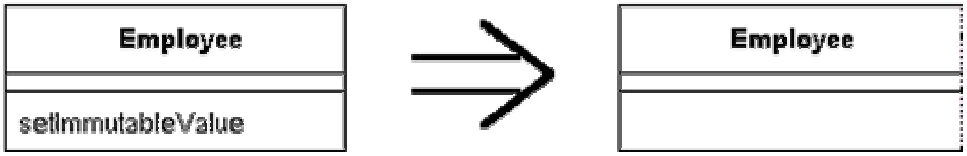

Figure 5-20: Remove Setting Method

5.1.18. RENAME METHOD

If the name of a method does not reveal its purpose, its name should be changed [3, 50]. However, the refactoring Rename Method can be generalized to rename variable, field, class, interface or package. Rename Method is illustrated in Figure 5-21. The method name is changed to be more readable and understandable to a human. Methods can be renamed to communicate their intention using this refactoring [3]. This refactoring makes methods names meaningful, self-explained and unambiguous and hence more maintainable [37] and usable [39]. Therefore, the refactoring "Rename Method" improves the *maintainability* and *usability* of the application framework.

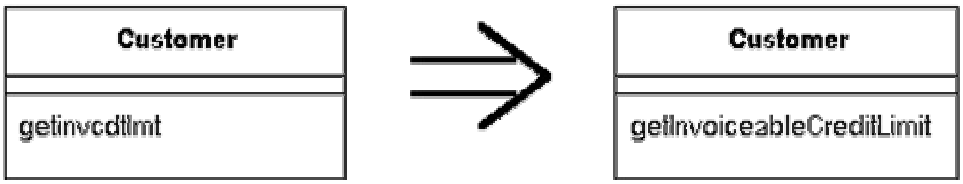

Figure 5-21: Rename Method

5.1.19. REPLACE CONDITIONAL WITH POLYMORPHISM

If a condition chooses a different behavior based on the type of an object, each clause of the condition is moved to an overriding method in a subclass and the original method is converted to abstract [3, 50]. The refactoring Replace Conditional with Polymorphism is exemplified in Figure 5-22. The explicit conditional is replaced by subclasses creation with different appropriate behaviors and members (fields and methods) [3]. This refactoring is an implementation of inheritance and abstraction,

essential architecture designs of frameworks [2, 4-6], to make the application more reusable [35], flexible and extensible [2]. It also reduces efforts for maintenance and update and decreases dependencies among components when adding new type to the conditional [3]. Thus, the refactoring "Replace Conditional with Polymorphism" improves the *reusability*, *maintainability*, *flexibility* and *extensibility* of the application framework.

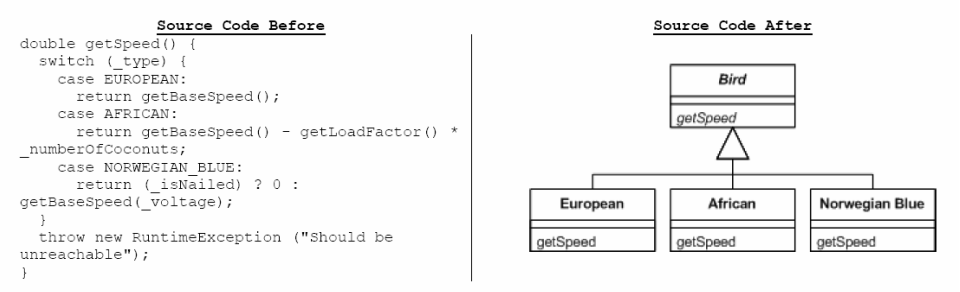

Figure 5-22: Replace Conditional with Polymorphism

5.1.20. REPLACE DELEGATION WITH INHERITANCE

When many simple delegations are used for the entire interface, then the delegating class can be extracted as a subclass of the delegate [3, 50]. This refactoring method is shown in Figure 5-23. All methods of the delegate class are used by inheriting the delegate class. This refactoring implements inheritance and abstraction, essential architecture designs of frameworks [2, 4-6] and makes the application more reusable [35], flexible and extensible [2]. Therefore, Extract Subclass improves the *reusability*, *flexibility* and *extensibility* of the application framework.

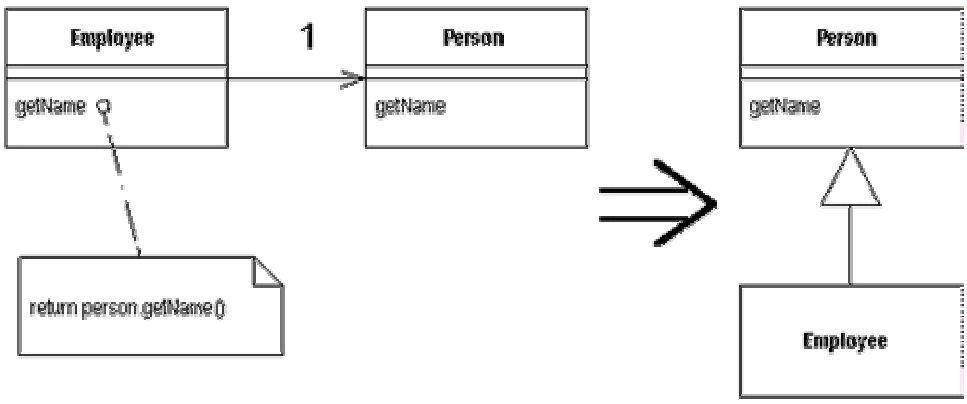

Figure 5-23: Replace Delegation with Inheritance

5.1.21. REPLACE MAGIC NUMBER WITH SYMBOLIC CONSTANT

If a literal number with a particular meaning exists, a constant is created with a meaningful name to replace the number [3, 50]. Figure 5-24 shows an example of this refactoring. A constant value is set in a constant variable. This refactoring allows to reuse and reference the symbolic constant in more than one place in the code [3]. It also reduces the difficulty of changing and maintaining the values [3]. It facilities exploring the logic and provides a great improvement in readability [3] and hence the maintainability [38]. Consequently, Replace Magic Number with Symbolic Constant improves the *reusability* and *maintainability* of the application framework.

Source Code Before

```
double potentialEnergy(double mass,
double height) {
 return mass * height * 9.81;
}
```

Source Code After

```
double potentialEnergy(double mass,
double height) {
 return mass *
GRAVITATIONAL_CONSTANT * height;
}
static final double
GRAVITATIONAL_CONSTANT = 9.81;
```

Figure 5-24: Replace Magic Number with Symbolic Constant

5.1.22. REVERSE CONDITIONAL

A condition, which would be easier to understand when it is reversed, should be reversed and reordered [50]. Reverse Conational is shown in Figure 5-25. This refactoring allows to re-phrase conditional statements in order to be more readable and understandable [50] and hence more maintainable [37] and usable [39]. Thus, Reverse Conditional improves the *maintainability* and *usability* of the application framework.

Source Code Before

```
if ( !isSummer( date ) )
 charge = winterCharge( quantity );
else
 charge = summerCharge( quantity );
```

Source Code After

```
if ( isSummer( date ) )
 charge = summerCharge( quantity );
else
 charge = winterCharge( quantity );
```

Figure 5-25: Reverse Conditional

5.1.23. SPLIT LOOP

If a loop does two jobs, the loop is duplicated per job [50]. Figure 5-26 illustrates the refactoring Split Loop. The complicated loop block is split into multiple simple loop blocks. This refactoring makes the loop blocks more clear [50]. It is an example of decomposing complex algorithms to be maintainable [37]. Therefore, the refactoring "Split Loop" improves the *maintainability* of the application frameworks.

Source Code Before

```
void printValues() {
 double averageAge = 0;
 double totalSalary = 0;
 for (int i = 0; i < people.length;
i++) {
   averageAge += people[i].age;
   totalSalary += people[i].salary;
 }
 averageAge = averageAge /
people.length;
 System.out.println(averageAge);
 System.out.println(totalSalary);
}
```

Source Code After

```
void printValues() {
 double totalSalary = 0;
 for (int i = 0; i < people.length;
i++) {
   totalSalary += people[i].salary;
 }
 double averageAge = 0;
 for (int i = 0; i < people.length;
i++) {
   averageAge += people[i].age;
 }
 averageAge = averageAge /
people.length;
 System.out.println(averageAge);
 System.out.println(totalSalary);
}
```

Figure 5-26: Split Loop

5.2. REFACTORINGS OF APPLICATION FRAMEWORK QUALITY ATTRIBUTES

This section summarizes the positive contributions of the selected refactoring methods on the application frameworks quality attributes discussed in the previous section. Table 5-1 shows the refactorings methods verses the quality attributes of application frameworks. The sign (√) is placed beside any refactoring method under quality attributes positively affected by the refactoring method. The empty cells represent that neither positive nor negative effects investigated.

Table 5-1: Refactorings and the Positive Effects on Quality

Refactoring Method	Reus.	Mod.	Maint.	Usab.	Flex.	Ext.
Add Parameter					√	√
Decompose Conditional	√		√			
Encapsulate Field	√	√	√			
Extract Method	√		√			
Extract Package					√	
Extract Subclass	√				√	√
Extract Super-Class	√				√	√
Hide Delegate	√	√	√			
Inline Class	√					
Move Field	√	√				
Parameterize Method					√	√
Pull Up Field	√					
Pull Up Method	√		√			
Push Down Field	√					
Remove Assignments to Parameters			√			
Remove Parameter	√					
Remove Setting Method	√					
Rename Method			√	√		
Replace Conditional with Polymorphism	√		√		√	√
Replace Delegation with Inheritance	√				√	√
Replace Magic Number with Symbolic Constant	√		√			
Reverse Conditional			√	√		
Split Loop			√			

5.2.1. REFACTORINGS OF REUSABILITY

One of the main features and benefits of application frameworks is reusability [2]. Reusable frameworks can be developed by generalizing existing applications [2]. Moreover, existing functionalities, or functional elements can be reused within the application [35, 52]. They can be reused by inheriting classes, overriding methods [2] and applying different levels of abstraction [2, 35]. The reusability of application framework is positively affected when applying a certain set of refactorings. The refactorings set includes *Decompose Conditional*, *Encapsulate Field*, *Extract Method*, *Extract Subclass*, *Extract Super-Class*, *Hide Delegate*, *Inline Class*, *Move Field*, *Pull Up Field*, *Pull Up*

Method, *Push Down Field*, *Remove Parameter*, *Remove Setting Method*, *Replace Conditional with Polymorphism*, *Replace Delegation with Inheritance* and *Replace Magic Number with Symbolic Constant*.

5.2.2. REFACTORINGS OF MODULARITY

One main feature of application frameworks is modularity [2]. The modularity can be enhanced by encapsulating implementation details behind interfaces [2]. The modularity of the application framework is enhanced by applying some refactorings. The modularity refactoring set consists of *Encapsulate Field*, *Hide Delegate* and *Move Field*.

5.2.3. REFACTORINGS OF MAINTAINABILITY

The application framework should be maintainable [2-4]. The maintainability of the application framework is improved by decomposing application source code blocks and elements into algorithms, blocks and elements [37]. Moreover, it is improved by applying certain refactorings. The refactoring set for maintainability includes *Decompose Conditional*, *Encapsulate Field*, *Extract Method*, *Hide Delegate*, *Pull Up Method*, *Remove Assignments to Parameters*, *Rename Method*, *Replace Conditional with Polymorphism*, *Replace Magic Number with Symbolic Constant*, *Reverse Conditional* and *Split Loop*.

5.2.4. REFACTORINGS OF USABILITY

The application framework should be usable [4]. The usability of the application framework can be improved using suitable and unambiguous error messages [41]. It is positively changed by applying certain refactoring methods. The refactoring methods for usability include *Rename Method* and *Reverse Conditional*.

5.2.5. REFACTORINGS OF FLEXIBILITY

The application framework can be developed by improving flexibility [2]. The flexibility can be accomplished by inheriting classes and overriding methods [2]. The flexibility of the application framework is increased by applying certain refactoring methods. The refactoring methods for flexibility include *Add Parameter*, *Extract Package*, *Extract Subclass, Extract Super-Class*, *Parameterize Method*, *Replace Conditional with Polymorphism* and *Replace Delegation with Inheritance*.

5.2.6. REFACTORINGS OF EXTENSIBILITY

The application frameworks can be developed by improving extensibility [2]. The extensibility can be accomplished by inheriting classes and overriding methods [2]. The extensibility of software is improved by applying certain refactorings. The extensibility refactorings set includes *Add Parameter*, *Extract Subclass*, *Extract Super-Class*, *Parameterize Method*, *Replace Conditional with Polymorphism* and *Replace Delegation with Inheritance*.

CHAPTER 6

REFACTORING TO FRAMEWORK

In this chapter, we present two approaches for refactoring to framework. The target of these approaches to produce the domain classes of existing applications to be used as framework. The first approach is based on quality attributes of application framework. The second approach is composed of a sequence of ordered refactoring methods and is based on the refactoring level or structure. These approaches are applied using two software applications and are also compared with each other. As an experimental procedure, we perform each refactoring to framework approach on two different applications from various areas and in different project size. These applications are SJEA from cryptography domain [53] and JFTP Client from network programming domain [54]. The source of these applications is Source Forge website [55].

Simple JAVA Encryption Algorithm (SJEA) is a simple command-line binary encryption algorithm of files (symmetric block cipher) written in JAVA. It uses a password and a byte-vector array to scramble the input file [53, 55] (http://sourceforge.net/projects/sjea/). SJEA application consists of 3 classes. The source code of SJEA is appended in Appendix A. On the other hand, JAVA File Transfer Protocol Client (JFTP) is a simple cross-platform ftp-client coded in JAVA with a command-line interface (CLI) and capable to support Graphical User Interface (GUI)

(http://sourceforge.net/projects/javaftp/) [54, 55]. JFTP application consists of 23 classes. The source code of SJEA is appended in Appendix B.

To demonstrate the changes done on each application during the phases of refactoring to framework, system-level results are presented using the software metrics: Line of Code (LOC), Number of Local Methods (NLM), Number of Classes (NOCL) and Number of Packages (NOP). LOC counts total number the physical lines of codes, NLM is the total number of methods with different scopes including private, protected, default and public. NOCL represents the number of classes of the software application and NOP is the number of packages in the software applications. To automate the process of software metrics analysis, we used "*Metamata*" metrics tool to collect the software metrics for the source code among the phases [56].

6.1. REFACTORING TO DOMAIN

It is important to design domain and application functionalities and their relationship with frameworks. The domain classes of an application can be defined as the classes that implement and provide functionalities of a certain domain area. On the other hand, the application classes of an application implement the specific application functionalities and behaviors. In this section, we propose a new refactoring process called Refactoring to Domain (RtD). The purpose of this process is to refactor existing classes of applications to domain classes by enhancing their object-oriented features such as inheritance and abstraction. The process encompasses a set of recommended steps to be

used, wherever there is a need, before starting the refactoring to framework approach. The benefit of refactoring to domain is to facilitate the refactoring to framework process by preparing the application's classes to be ready for refactoring to framework.

The steps of Refactoring to Domain are as follows:

- If there is a class contains both domain and application functionality, a new class is created to contain the application functionality and the source class becomes a pure domain class. However, the domain functionality provides a common core service in the domain such as encryption and decryption algorithm, FTP commands actions, IO (input and output) buffering, etc. The application functionality provides a specific-application behavior such as the way of receiving request and displaying response.
- If the domain functionalities of a class are defined as static methods, the static methods are replaced with dynamic ones to support object-oriented abstraction which is an essential architecture design of frameworks [2, 4-6]. In order to do that, the common parameters of static methods are removed and replaced with local variables and their references are replaced too. Constructors with different signatures are created according to the new local variables created. Instances of the objects are used in replacing the references of the removed parameters of the static methods.
- If a domain class is final class, the final keyword is removed to allow class extension or inheritance.

6.1.1. REFACTORING SJEA TO DOMAIN

The SJEA application originally consists of three classes; Checksum, Decryption and Encryption. All of the three classes contain domain and application functionalities. The domain functions are the methods responsible for creating checksum, decryption and encryption. The application functions are the main methods in each class providing the command-line interface of each class.

To refactor SJEA to domain, we first create a new class for the application functions (main methods) in order to separate the application class from the domain classes. Consequently, the domain classes SJEA are Checksum, Decryption and Encryption while the application class is SJEAApplication. Since the domain functions of the SJEA classes are originally implemented as static methods, they should be converted to dynamic methods. To perform this, the common parameter of the static methods (file) is replaced with a local variable and two constructors are created; default or empty constructor and parameterized constructor. As a result of applying refactoring to domain, the SJEA application is ready to be refactored to framework.

6.1.2. REFACTORING JFTP TO DOMAIN

The JFTP design distinguishes and separates the domain classes from the application classes. However, it has a domain class named FTPCmdServer that includes all FTP command services functions. This class is final. The final class in java cannot be

inherited or extended. The final keyword is removed as a step of refactoring to domain. Therefore, the JFTP application becomes ready to be refactored to framework.

6.2. QUALITY ATTRIBUTE BASED REFACTORING TO FRAMEWORK (QARTF)

In this section, we propose a quality attribute based approach refactoring to framework. It can be followed to improve the domain classes of an existing application to be used as framework. The quality attribute based refactoring to framework (QARtF) mainly focuses on improving the framework quality attributes of an application using the refactorings identified in Chapter 5. It consists of three ordered phases: reusability and modularity, maintainability and usability and flexibility and extensibility. It merges the reusability and modularity phase because it contains the large set of refactoring methods and the reusability is improved when the modularity is improved [32]. Then, it merges the maintainability and usability in one phase because they share the same refactoring methods. It merges the flexibility and extensibility in the last phase because they cover same set of refactorings. We structure the phases based on the number of refactoring methods per phase in descending order. The reason of this structuring order is to apply maximum number of refactoring methods and to cover overlapping refactorings in early phases of the approach. As a suggested recommendation, meaningful code statements and clear comments should be used in each step of the phases of QARtF. However, the refactoring methods of each phase of QARtF are not ordered.

6.2.1. REUSABILITY AND MODULARITY REFACTORING

Reusability and modularity refactoring is the first phase of QARtF approach. It encompasses refactoring methods that positively affect the reusability and modularity of an application. There are 16 refactorings in the usability and modularity phase. The refactorings set includes *Decompose Conditional*, *Encapsulate Field*, *Extract Method*, *Extract Subclass*, *Extract Super-Class*, *Hide Delegate*, *Inline Class*, *Move Field*, *Pull Up Field*, *Pull Up Method*, *Push Down Field*, *Remove Parameter*, *Remove Setting Method*, *Replace Conditional with Polymorphism*, *Replace Delegation with Inheritance* and *Replace Magic Number with Symbolic Constant*.

6.2.2. MAINTAINABILITY AND USABILITY REFACTORING

Maintainability and usability refactoring is the second phase of QARtF approach. It covers refactorings that improve the maintainability and usability of an application. There are 11 refactorings methods in the maintainability and usability phase. The refactorings of the maintainability and usability phase are *Decompose Conditional*, *Encapsulate Field*, *Extract Method*, *Hide Delegate*, *Pull Up Method*, *Remove Assignments to Parameters*, *Rename Method*, *Replace Conditional with Polymorphism*, *Replace Magic Number with Symbolic Constant*, *Reverse Conditional* and *Split Loop*.

6.2.3. FLEXIBILITY AND EXTENSIBILITY REFACTORING

Flexibility and extensibility refactoring is the last phase of QARtF approach. It includes a set of refactorings that positively affect the flexibility and extensibility of an application. There are 7 refactoring methods in this phase. The refactoring methods of the flexibility and extensibility phase are *Add Parameter*, *Extract Package*, *Extract Subclass, Extract Super-Class*, *Parameterize Method*, *Replace Conditional with Polymorphism* and *Replace Delegation with Inheritance*.

6.2.4. MEANINGFUL CODING AND CLEAR COMMENTING

The maintainability of an application is improved when its source code has meaningful components identification and names and understandable statement, commands and understandable comments [37]. This step is applicable in any phase of QARtF. *Rename Method*, *Field*, *Variable*, *Class*, *Interface* and *Package* refactorings are helpful and useful in this step.

6.2.5. REFACTORING SJEA TO FRAMEWORK USING QARTF

The SJEA application is refactored to framework using the QARtF approach in three phases. First, the reusability and modularity refactorings are applied on each code smells of the SJEA application. Second, all of applicable maintainability and usability refactorings are made on the SJEA classes. Finally, the flexibility of the SJEA is

improved by applying the last phase of QARtF. In each phase QARtF, meaningful coding and clear commenting step is used.

In refactoring SJEA to a framework, most of the refactoring methods of the phases of QARtF are applied. The reusability and modularity refactoring phase includes *Encapsulate Field*, *Extract Method*, *Extract Subclass*, *Extract Super-Class*s, *Hide Delegate*, *Pull Up Field*, *Pull Up Method*, *Remove Setting Method*, *Replace Magic Number with Symbolic Constant* and *Replace Delegation with Inheritance*. In the maintainability and usability phase, most of the refactorings are already covered by the reusability and modularity refactoring methods. However, the refactoring methods *Rename Method*, *Reverse Conditional* and *Split Loop* are not applied on SJEA due to the absence of their code smells. Finally, the flexibility and extensibility phase is applied on the SJEA application using the refactoring method *Extract Package*. Table 6-2 shows the refactoring methods of QARtF applied on SJEA classes.

Table 6-2: SJEA Classes Refactoring Using QARtF

Class	Type	Refactorings Applied (Phase Number)
Checksum	Domain	Encapsulate Field (1), Remove Setting Method (1), Replace Magic Number with Symbolic Constant (1), Extract Method (1), Extract Package (3)
ChecksumMD5	Domain	Extract Subclass (1), Extract Package (3)
Cryptography	Domain	Extract Super-Class (1), Pull Up Method (1), Replace Delegation with Inheritance (1), Extract Package (3)
Decryption	Domain	Encapsulate Field (1), Remove Setting Method (1), Hide Delegate (1), Extract Method (1), Extract Package (3)
Encryption	Domain	Encapsulate Field (1), Remove Setting Method (1), Hide Delegate (1), Extract Method (1), Extract Package (3)
SJEAApplication	Application	N/A

The results, in terms of system-level LOC, NLM, NOCL and NOP, of each phase of the QARtF approach are presented in Table 6-3. It shows that there is no change accomplished in the maintainability and usability phase because its refactoring methods are already covered by the first phase or are not applicable in any code smell. It also shows increment in NLM, NOCL and NOP. NLM is increased due to methods extraction during the refactoring to framework phases. NOCL is increased because of classes' extraction as new subclasses and super-classes are created. NOP is increased because packages are extracted to group related classes together.

Table 6-3: Results of SJEA Refactoring Using QARtF

Phase	LOC	NLM	NOCL	NOP
SJEA Application	334	7	4	1
Reusability and Modularity of SJEA	328	11	6	1
Maintainability and Usability of SJEA	328	11	6	1
Flexibility and Extensibility of SJEA	331	11	6	3
SJEA Framework	331	11	6	3

6.2.6. REFACTORING JFTP TO FRAMEWORK USING QARTF

The JFTP application is refactored to framework via the phases of the QARtF approach. First, the reusability and modularity refactorings are applied on each code smells of the JFTP application. Second, all of applicable maintainability and usability refactorings are applied on the JFTP classes. Finally, the flexibility and extensibility of the JFTP are improved by applying the flexibility and extensibility refactorings. In each phase of QARtF, meaningful coding and clear commenting step is used.

In refactoring JFTP to a framework, all of the refactoring methods of the QARtF phases are applied. The reusability and modularity refactoring phases includes applying *Decompose Conditional*, *Encapsulate Field*, *Extract Method*, *Extract Subclass*, *Extract Super-Class*, *Hide Delegate*, *Inline Class*, *Move Field*, *Pull Up Field*, *Pull Up Method*, *Push Down Field*, *Remove Parameter*, *Remove Setting Method*, *Replace Conditional with Polymorphism*, *Replace Delegation with Inheritance* and *Replace Magic Number with Symbolic Constant*. In the next phase, the maintainability and usability refactorings applied are *Remove Assignments to Parameters*, *Rename Method*, *Reverse Conditional* and *Split Loop*. Finally, the flexibility and extensibility phase is applied in the JFTP application using the refactoring methods: *Add Parameter*, *Extract Package* and *Parameterize Method*. Table 6-4 shows the refactoring methods of QARtF applied on the JFTP application's classes.

Table 6-4: JFTP Classes Refactoring Using QARtF

Class	Type	Refactorings Applied (Phase Number)
Authenticateable	FTP Domain	Inline Class (1), Extract Package (3)
CommandLineParser	FTP Domain	Decompose Conditional (1), Hide Delegate (1), Remove Assignments to Parameters (2), Rename Method (2), Add Parameter (3), Extract Package (3)
Connectable	FTP Domain	Inline Class (1), Extract Package (3)
Createable	FTP Domain	Inline Class (1), Extract Package (3)
DataTypeChangeable	FTP Domain	Extract Subclass (1), Inline Class (1), Extract Package (3)
DirectoryChangeable	FTP Domain	Extract Subclass (1), Inline Class (1), Extract Package (3)
FTPASCIIData	FTP Domain	Decompose Conditional (1), Replace Conditional with Polymorphism (1), Extract Package (3)
FTPAuthentication	FTP Domain	Decompose Conditional (1), Extract Subclass (1), Push Down Field (1), Extract Package (3)
FTPBinaryData	FTP Domain	Decompose Conditional (1), Replace Conditional with Polymorphism (1), Extract Package (3)
FTPCmdServer	FTP Domain	Encapsulate Field (1), Remove Setting Method (1), Extract Package (3)
FTPConnection	FTP Domain	Decompose Conditional (1), Extract Method (1), Extract Subclass (1), Extract Super-Class (1), Hide Delegate (1), Pull Up Method (1), Push Down Field (1), Remove Parameter (1), Remove Assignments to Parameters (2), Extract Package (3), Parameterize Method (3)
FTPDataType	FTP Domain	Extract Method (1), Extract Subclass (1), Replace Magic Number with Symbolic Constant (1), Extract Package (3)
FTPDirectory	FTP Domain	Decompose Conditional (1), Extract Subclass (1), Hide Delegate (1), Push Down Field (1), Reverse Conditional (2), Split Loop (2), Extract Package (3)
FTPTransfer	FTP Domain	Decompose Conditional (1), Extract Subclass (1), Hide Delegate (1), Push Down Field (1), Replace Delegation with Inheritance (1), Extract Package (3)
JFTP	Application	Decompose Conditional (1), Extract Method (1), Extract Subclass (1), Move Field (1), Remove Assignments to Parameters (2)
JFTPSuper	FTP Domain	Extract Package (3)
Listable	FTP Domain	Inline Class (1), Extract Package (3)
NetIO	IO Domain	Encapsulate Field (1), Inline Class (1), Pull Up Field (1), Pull Up Method (1), Extract Package (3)
NetReader	IO Domain	Extract Package (3)
NetWriter	IO Domain	Move Field (1), Extract Package (3)
Parser	FTP Domain	Extract Package (3)
Progressbar	Application	Extract Package (3)
Removeable	FTP Domain	Inline Class (1), Extract Package (3)
ServerResponseParser	FTP Domain	Extract Package (3)
StdErr	IO Domain	Inline Class (1), Extract Package (3)
StdIn	IO Domain	Inline Class (1), Extract Package (3)
StdOut	IO Domain	Inline Class (1), Move Field (1), Extract Package (3)
Transferable	FTP Domain	Inline Class (1), Extract Package (3)

The results, in terms of system-level LOC, NLM, NOCL and NOP, of each phase of the QARtF approach are presented in Table 6-5. It illustrates the changes of the JFTP application accomplished in each phase of QARtF. It also shows increment in NLM, NOCL and NOP. NLM is increased due to methods extraction and decomposition during the refactoring to framework phases. NOCL is increased because of classes' extraction and abstraction as new subclasses and super-classes are created. NOP is increased because packages are extracted to group relevant classes together.

Table 6-5: Results of JFTP Refactoring Using QARtF

Phase	LOC	NLM	NOCL	NOP
JFTP Application	1490	180	23	1
Reusability and Modularity of JFTP	1711	256	28	1
Maintainability and Usability of JFTP	1817	281	28	1
Flexibility and Extensibility of JFTP	1865	283	28	6
JFTP Framework	1865	283	28	6

6.3. LEVEL-BASED REFACTORING TO FRAMEWORK (LRTF)

In this section, a level-based approach of refactoring to framework is defined. The level-based refactoring to framework (LRtF) consists of a sequence of ordered refactoring methods. The refactoring methods of the level-based refactoring to framework include all refactorings of the quality attribute based refactoring to framework (QARtF). It covers all refactorings which improve and enhance the application framework quality attributes. In the level-based refactoring to framework, we classify and organize the refactorings in a sequential approach. They are classified into five different levels: package level, class level, method level, field level and block level. The block level is divided into loop level and if-clause level. The level classification identifies the type of changes that a refactoring

does on a source code. It can be used to avoid overlapping and conflicting between the refactoring methods. Table 6-6 summarizes the steps of the level-based refactoring to framework (LRtF).

The package-level refactorings set includes only *Extract Package*. In class-level refactoring, there are six refactoring methods: *Inline Class*, *Extract Subclass*, *Extract Super-Class*, *Replace Conditional with Polymorphism*, *Hide Delegate* and *Replace Delegation with Inheritance*. The class-level refactorings cover different types of operations on classes: remove, extract, abstract and delegate. The field-level refactoring methods encompass *Replace Magic Number with Symbolic Constant*, *Encapsulate Field*, *Push Down Field*, *Move Field* and *Pull Up Field*. The method-level refactorings are *Extract Method*, *Decompose Conditional*, *Remove Setting Method*, *Pull Up Method*, *Remove Parameter*, *Rename Method*, *Add Parameter*, *Parameterize Method* and *Remove Assignments to Parameters*. *Split Loop* is identified as a loop-level refactoring in block-level. *Reverse Conditional* is recognized as if-clause-level refactoring methods.

Table 6-6: Level-Based Refactoring to Framework (LRtF)

	Refactoring	Level
1	Extract Subclass	Class
2	Extract Super-Class	Class
3	Replace Conditional with Polymorphism	Class
4	Hide Delegate	Class
5	Replace Delegation with Inheritance	Class
6	Inline Class	Class
7	Extract Package	Package
8	Replace Magic Number with Symbolic Constant	Field
9	Encapsulate Field	Field
10	Push Down Field	Field
11	Move Field	Field
12	Pull Up Field	Field
13	Extract Method	Method
14	Decompose Conditional	Method
15	Remove Setting Method	Method
16	Pull Up Method	Method
17	Remove Parameter	Method
18	Add Parameter	Method
19	Parameterize Method	Method
20	Remove Assignments to Parameters	Method
21	Rename Method	Method
22	Reverse Conditional	If Clause
23	Split Loop	Loop

6.3.1. CLASS-LEVEL REFACTORINGS

The class-level refactorings should be applied first because classes determine the size of the application in terms of number of classes in early stage of the refactoring to framework. The class-level refactorings are performed according to extract refactorings, abstract refactorings, delegate refactorings and remove refactorings. First, All possible subclasses are created using *Extract Subclass*. *Extract Subclass* should be done first because some of the extracted subclasses can be refactored using next class-level refactorings. Then, the common features of classes can be combined in a super-class using *Extract Super-Class* refactoring. *Extract Super-Class* refactoring can be applied on the

refactored subclasses resulted by *Extract Subclass*. Next, the abstract refactoring is applied using *Replace Conditional with Polymorphism* method. The type conditional expression should be changed to polymorphism using the refactoring method *Replace Conditional with Polymorphism*. This improves inheritance and abstraction levels of the application. *Hide Delegate* and *Replace Delegation with Inheritance* are delegate refactorings and applied after abstract refactoring. This is to finalize the inheritance and abstraction level of the classes. *Hide Delegate* is used to hide all delegates of classes. Then, *Replace Delegation with Inheritance* refactoring method may be used on existing delegates and hidden delegates to be replaced by inheritance. This increases the level of abstraction and inheritance. Finally at class-level refactoring, all unnecessary classes and interfaces should be removed through *Inline Class* refactoring method. The class-level refactorings fix the size of the application in terms of the number of classes and finalize the level of abstraction and inheritance.

6.3.2. PACKAGE-LEVEL REFACTORINGS

Since the application size is fixed and the number of classes and interfaces are determined in the class-level refactorings phase, it is easy to group classes into different scopes or packages based on their functionalities. Different packages can be extracted using the refactoring *Extract Package*.

6.3.3. FIELD-LEVEL REFACTORINGS

The field-level refactorings are done before the method-level refactoring because some field-level refactorings cause creating new methods and these new methods may need to be refactored using the method-level refactorings. The field-level refactoring starts with *Replace Magic Number with Symbolic Constant* because the symbolic constant can be replace by a field in a class which may need some other field-level refactorings. Then, all fields of a class should be encapsulated in getter and setter methods via the refactoring method *Encapsulate Field*. After field encapsulation, the specialized fields of a super-class are pushed down with their encapsulations to appropriate subclasses using *Push Down Field*. *Move Field* refactoring is useful to transfer certain fields with their encapsulations from a class to another. The common original or refactored fields are pulled up with their encapsulations to a super-class by the refactoring method *Pull Up Field*.

6.3.4. METHOD-LEVEL REFACTORINGS

The method-level refactoring can begin with *Extract Method* as a kind of decomposing complex algorithms. Complex conditional can be extracted to a new method using *Decompose Conditional* refactoring. Unnecessary encapsulation methods can be removed through the refactoring method *Remove Setting Method*. Then, the exiting, extracted or decomposed methods can be moved from a subclass to a super-class using *Pull Up Method* refactoring. The external structures of classes are finalized by the refactoring *Pull Up Method*. After that, all needless parameters of methods should be

removed by the refactoring *Remove Parameter*. *Add Parameter* refactoring is used to add parameters to methods to increase the flexibility of methods. Sometimes, a method is more useful if it is overloaded with different signature to have parameters using the refactoring *Parameterize Method*. However, method parameters should be read only by applying *Remove Assignments to Parameters* refactoring. *Rename Method* refactoring can be used to make the method reveal its purpose. The method-level refactorings finalize and fix total number of methods of an application.

6.3.5. IF-CLAUSE-LEVEL REFACTORINGS

The block-level refactoring comes at the end of the method-level refactoring because the method-level refactoring finalizes the number of methods of classes. The if-clause-level refactorings should come before the loop-level refactoring. This is because the loop-level results code duplication because it splits a single loop into multiple loops. If the split loop contains a condition that needs some if-clause refactorings, then starting with the if-clause-level refactorings reduces the amount of refactoring work at loop-level refactoring. Therefore, it is better to do if-clause-level refactorings before loop-level refactorings. When a condition expression is needed to be reversed to be more readable and understandable, the if-clause-level refactoring *Reverse Conditional* is applied.

6.3.6. LOOP-LEVEL REFACTORINGS

The loop-level refactoring is applicable after the if-clause-level refactoring. *Split Loop* refactoring method is used to decompose a complex loop into several duplicated simple loops.

6.3.7. NAME AND COMMENT REFACTORINGS

The naming refactoring methods can be used in any phase of the level-based refactoring to framework without any negative effects on other refactorings. The naming refactorings includes *Rename Method*, *Rename Variable*, *Rename Field*, *Rename Class*, *Rename Interface* and *Rename Package*. The main advantage of the naming refactoring is to make method, variable, field, class, interface, or package more readable and more understandable. In addition, meaningful comments on source code statements, explaining the input and output specifications, help and improve the readability of the source code statements. They can be done using *Add Comments*. However, this phase can be done among all of the other phases of LRtF.

6.2.5. REFACTORING SJEA TO FRAMEWORK USING LRTF

The SJEA application is refactored to framework using the LRtF approach. First, the class-level refactorings are applied on each code smells of the SJEA application. Then, the package-level refactoring is made on the SJEA classes. The field-level refactorings, after that, are applied on the SJEA application. Some method-level refactorings are also

done on the SJEA classes. In each phase LRtF, meaningful coding and clear commenting are used.

In refactoring SJEA to a framework, most of the refactoring methods of the phases of LRtF are applied. The class-level refactorings include *Extract Subclass*, *Extract Super-Clas*s, *Hide Delegate* and *Replace Delegation with Inheritance*. The package-level refactoring is *Extract Package*. The field-level refactorings applied are *Encapsulate Field*, *Pull Up Field* and *Replace Magic Number with Symbolic Constant*. The method-level refactorings applied include *Extract Method*, *Pull Up Method* and *Remove Setting Method*. The SJEA application does not have code smells for the if-clause-level and loop-level refactorings. Table 6-7 shows the refactoring methods of LRtF applied on SJEA classes.

Table 6-7: SJEA Classes Refactoring Using LRtF

Class	Type	Refactorings Applied (Phase Number)
Checksum	Domain	Encapsulate Field (3), Remove Setting Method (4), Replace Magic Number with Symbolic Constant (3), Extract Method (4), Extract Package (2)
ChecksumMD5	Domain	Extract Subclass (1), Extract Package (2)
Cryptography	Domain	Extract Super-Class (1), Pull Up Method (4), Replace Delegation with Inheritance (1), Extract Package (2)
Decryption	Domain	Encapsulate Field (3), Remove Setting Method (4), Hide Delegate (1), Extract Method (4), Extract Package (2)
Encryption	Domain	Encapsulate Field (3), Remove Setting Method (4), Hide Delegate (1), Extract Method (4), Extract Package (2)
SJEAApplication	Application	N/A

The system-level results, in terms of LOC, NLM, NOCL and NOP, of each phase of the LRtF approach are presented in Table 6-8. It shows that there is no change accomplished in if-clause-level and loop-level phases because there are no suitable code smells in the SJEA. It presents increment in NLM, NOCL and NOP. NLM is increased

due to methods extraction during LRtF phases. NOCL is increased because of classes' extraction as new subclasses and super-classes are created. NOP is increased because packages are extracted to group relevant classes together.

Table 6-8: Results of SJEA Refactoring Using LRtF

Phase	LOC	NLM	NOCL	NOP
SJEA Application	334	7	4	1
Class-Level of SJEA	314	5	6	1
Package-Level of SJEA	317	5	6	3
Field-Level of SJEA	331	11	6	3
Method-Level of SJEA	331	11	6	3
If-Clause-Level of SJEA	331	11	6	3
Loop-Level of SJEA	331	11	6	3
SJEA Framework	331	11	6	3

6.2.6. REFACTORING JFTP TO FRAMEWORK USING LRTF

The JFTP application is refactored to framework via the phases of the LRtF approach. First, the class-level refactorings are applied on each code smells of the JFTP application. Then, the package-level refactoring is made on the JFTP classes. The field-level refactorings, after that, are applied on the JFTP application. The method-level refactorings are also done on the JFTP classes. Finally, the block-level refactorings (if-clause-level and loop-level) are applied. Meaningful coding and clear commenting are used while refactoring the JFTP application in each phase LRtF.

In refactoring JFTP to a framework using LRtF, all of the refactoring methods of the LRtF phases are applied. The class-level refactorings include *Inline Class*, *Extract Subclass*, *Extract Super-Class*, *Replace Conditional with Polymorphism*, *Hide Delegate*

and *Replace Delegation with Inheritance*. The package-level refactoring is *Extract Package*. The field-level refactorings applied are *Replace Magic Number with Symbolic Constant*, *Encapsulate Field*, *Push Down Field*, *Move Field* and *Pull Up Field*. The method-level refactorings applied includes *Extract Method*, *Decompose Conditional*, *Remove Setting Method*, *Pull Up Method*, *Remove Parameter*, *Add Parameter*, *Parameterize Method*, *Remove Assignments to Parameters*, and *Rename Method*. The loop-level refactoring applied is *Split Loop* and the if-clause-level refactoring applied is *Reverse Conditional*. Table 6-9 shows the refactoring methods of LRtF applied on JFTP classes.

Table 6-9: JFTP Classes Refactoring Using LRtF

Class	Type	Refactorings Applied (Phase Number)
Authenticateable	FTP Domain	Inline Class (1), Extract Package (2)
CommandLineParser	FTP Domain	Decompose Conditional (4), Hide Delegate (1), Remove Assignments to Parameters (4), Rename Method (4), Add Parameter (4), Extract Package (2)
Connectable	FTP Domain	Inline Class (1), Extract Package (2)
Createable	FTP Domain	Inline Class (1), Extract Package (2)
DataTypeChangeable	FTP Domain	Extract Subclass (1), Inline Class (1), Extract Package (2)
DirectoryChangeable	FTP Domain	Extract Subclass (1), Inline Class (1), Extract Package (2)
FTPASCIIData	FTP Domain	Decompose Conditional (4), Replace Conditional with Polymorphism (1), Extract Package (2)
FTPAuthentication	FTP Domain	Decompose Conditional (4), Extract Subclass (1), Push Down Field (3), Extract Package (2)
FTPBinaryData	FTP Domain	Decompose Conditional (4), Replace Conditional with Polymorphism (1), Extract Package (2)
FTPCmdServer	FTP Domain	Encapsulate Field (3), Remove Setting Method (4), Extract Package (2)
FTPConnection	FTP Domain	Decompose Conditional (4), Extract Method (4), Extract Subclass (1), Extract Super-Class (1), Hide Delegate (1), Pull Up Method (4), Push Down Field (3), Remove Parameter (4), Remove Assignments to Parameters (4), Extract Package (2), Parameterize Method (4)
FTPDataType	FTP Domain	Extract Method (4), Extract Subclass (1), Replace Magic Number with Symbolic Constant (3), Extract Package (2)
FTPDirectory	FTP Domain	Decompose Conditional (4), Extract Subclass (1), Hide Delegate (1), Push Down Field (3), Reverse Conditional (5), Split Loop (6), Extract Package (2)
FTPTransfer	FTP Domain	Decompose Conditional (4), Extract Subclass (1), Hide Delegate (1), Push Down Field (3), Replace Delegation with Inheritance (1), Extract Package (2)
JFTP	Application	Decompose Conditional (4), Extract Method (4), Extract Subclass (1), Move Field (3), Remove Assignments to Parameters (4)
JFTPSuper	FTP Domain	Extract Package (2)
Listable	FTP Domain	Inline Class (1), Extract Package (2)
NetIO	IO Domain	Encapsulate Field (3), Inline Class (1), Pull Up Field (3), Pull Up Method (4), Extract Package (2)
NetReader	IO Domain	Extract Package (2)
NetWriter	IO Domain	Move Field (3), Extract Package (2)
Parser	FTP Domain	Extract Package (2)
Progressbar	Application	Extract Package (2)
Removeable	FTP Domain	Inline Class (1), Extract Package (2)
ServerResponseParser	FTP Domain	Extract Package (2)
StdErr	IO Domain	Inline Class (1), Extract Package (2)
StdIn	IO Domain	Inline Class (1), Extract Package (2)
StdOut	IO Domain	Inline Class (1), Move Field (3), Extract Package (2)
Transferable	FTP Domain	Inline Class (1), Extract Package (2)

The system-level results are presented in terms of LOC, NLM, NOCL and NOP for each phase of the LRtF approach in Table 6-10. It illustrates the changes of the JFTP application accomplished in each phase of LRtF. It shows increment occurred on NLM, NOCL and NOP. NLM is increased due to methods extraction and decomposition during the refactoring to framework phases. NOCL is increased because of classes' extraction and abstraction as new subclasses and super-classes are created. NOP is increased because packages are extracted to group related classes together.

Table 6-10: Results of JFTP Refactoring Using LRtF

Phase	LOC	NLM	NOCL	NOP
JFTP Application	1490	180	23	1
Class-Level of JFTP	1577	191	28	1
Package-Level of JFTP	1611	191	28	6
Field-Level of JFTP	1754	223	28	6
Method-Level of JFTP	1861	283	28	6
If-Clause-Level of JFTP	1861	283	28	6
Loop-Level of JFTP	1865	283	28	6
JFTP Framework	1865	283	28	6

6.4. CHARACTERISTICS OF QARTF AND LRTF APPROACHES

In this section, we state the main features and characteristics of the two refactoring to frameworks approaches: the Quality Attribute Based Refactoring to Framework (QARtF) and Level-Based Refactoring to Framework (LRtF). Although they share common features, each one of them has its own characteristics. Both approaches conclude with the same set of domain classes and application classes, as illustrated in Table 6-11 and 6-12. Appendix A and B contain the refactored source code of the applications. They both have the same application framework quality attributes improvement in the system

level. However, QARtF is a general refactoring to framework approach while LRtF is a standard and specific approach. QARtF is ordered phase level while LRtF provides sequential ordered refactoring methods. QARtF requires designer or developer experience with quality attributes. However, LRtF does not rely on designer or developer experience quality attribute because it is a sequence of refactoring methods.

Table 6-11: SJEA Classes after Refactoring to Framework

Class	Type	Description
Checksum	Domain	A Class implements checksum creation
ChecksumMD5	Domain	A Class implements checksum creation using MD5 algorithm
Cryptography	Domain	A Class implements cryptography of MD5 algorithm
Decryption	Domain	A Class implements decryption using MD5 algorithm
Encryption	Domain	A Class implements encryption using MD5 algorithm
SJEAApplication	Application	A Class implements the application user interface operation

Table 6-12: JFTP Classes after Refactoring to Framework

Class	Type	Description
Connectable	FTP Domain	An Interface includes methods for FTP connection
Authenticateable	FTP Domain	An Interface includes methods for FTP authentication
Listable	FTP Domain	An Interface includes methods for FTP directory list
DataTypeChangeable	FTP Domain	An Interface includes methods for FTP data mode change
DirectoryChangeable	FTP Domain	An Interface includes methods for FTP directory change
Createable	FTP Domain	An Interface includes methods for FTP directory creation
Removeable	FTP Domain	An Interface includes methods for FTP directory remove
Transferable	FTP Domain	An Interface includes methods wrapping all FTP data transfer
JFTPSuper	FTP Domain	A Super-Class of all FTP activities classes
FTPCmdServer	FTP Domain	A Super-Class of FTP activities and commands classes
FTPConnection	FTP Domain	A Class implements FTP connection activities
FTPAuthentication	FTP Domain	A Class implements FTP authentication activities
FTPDataType	FTP Domain	A Class implements FTP data mode change activities
FTPASCIIData	FTP Domain	A Class implements FTP data type change to ASCII
FTPBinaryData	FTP Domain	A Class implements FTP data type change to Binary or Image
FTPDirectory	FTP Domain	A Class implements FTP directory manipulation activities
FTPTransfer	FTP Domain	A Class implements FTP data transfer activities
Parser	FTP Domain	A Super-Class of all FTP messages parsing classes
CommandLineParser	FTP Domain	A Class implements FTP command parsing for requests
ServerResponseParser	FTP Domain	A Class implements FTP command parsing for responses
NetIO	IO Domain	A Class implements network IO operations
NetReader	IO Domain	A Class implements network reading operation
NetWriter	IO Domain	A Class implements network writing operation
StdErr	IO Domain	A Class implements standard error reporting operation
StdIn	IO Domain	A Class implements standard input reporting operation
StdOut	IO Domain	A Class implements standard output reporting operation
JFTP	Application	A Class implements the application user interface operation
Progressbar	Application	A Class implements progress bar for interface operation

CHAPTER 7

CONCLUSION

This chapter concludes the thesis with an overview of major contributions and directions for future work.

7.1. MAJOR CONTRIBUTIONS

In conclusion, the refactoring to framework is software refactoring or restructuring to produce reusable domain classes for a specific problem domain that can be used as a common application framework. In this work we proposed two refactoring to framework approaches consisting of a set of refactoring methods. The two refactoring to framework approaches provide a standard approach for application frameworks development using refactoring. In particular, the following contributions have been made:

1. Identified the software quality goals of the application frameworks.
2. Specified the refactoring methods that positively contribute to the quality goals of the application frameworks.
3. Proposed and composed two refactoring to framework approaches:
    a. Quality Attribute Based Refactoring to Framework.
    b. Level-Based Refactoring to Framework.
4. Experimentally applied the refactoring to framework approaches using existing software applications.

7.2. FUTURE WORK

This section presents additional research directions to be explored in future work. The future work can be summarized as follows:

- Apply the refactoring to framework approaches in many applications from diverse domain areas in different scales.
- Enhance the refactoring to framework approaches to include more refactoring methods that positively affect quality attributes of frameworks.
- Validate more refactoring methods on different quality attributes such as scalability, changeability and others.
- Investigate more refactoring methods and their effects on the application frameworks quality attributes.
- Develop a refactoring process approach to convert a set of existing applications to a product line.

APPENDIX A: SJEA APPLICATION SOURCE CODE

Appendix A presents the source code of the SJEA application's classes before and after applying the refactoring to frameworks. . It includes the original source code of the application and the resulted source codes after applying Quality Attribute Based Refactoring to Framework (QARtF) and Level-Based Refactoring to Framework (LRtF). Source Code A-1 contains SJEA original source code. Source Code A-2 contains SJEA original source Code after applying Refactoring to Domain (RtD). Source Code A-3 contains SJEA refactored source code using QARtF. Source Code A-4 contains SJEA refactored source code using LRtF.

```
package sjea.statics;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Checksum.java" - checksum verification tool
//  usage: java Checksum <decrypted-file> <checksum-file>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.security.MessageDigest;
import java.util.Scanner;

public class Checksum {

	public static int check(String filename1, String filename2) {
		int rc = 0;
		try {
			byte[] chk1 = createChecksum(filename1);
			BufferedReader	in	=	new	BufferedReader(new
FileReader(filename2));
			String str = "";
			str = in.readLine();
			in.close();
			String md5result1 = "";
			for (int i = 0; i < chk1.length; i++) {
				// prints checksum in hex
				md5result1  +=  Integer.toString((chk1[i]  &  0xff)  +
0x100, 16)
					.substring(1);
			}
			System.out.println("\n\n " + str + " < " + filename1);
			System.out.println("\n " + md5result1 + " < " + filename2);
			// check if two checksums match
			if (str.equals(md5result1)) {
				System.out.println("\n\nDONE! Checksum match!");
				rc = 1;
			} else {
				System.out.println("\n\nERROR!	Checksum	do	not
match!");
				rc = 2;
			}
			in.close();
			return rc;
		} catch (Exception e) {
			e.printStackTrace();
			return rc;
		}
	}
```

```
	public static byte[] createChecksum(String filename) throws Exception {
		// create checksum (MessageDigest), write using buffer
		InputStream fis = new FileInputStream(filename);
		byte[] buffer = new byte[2000000];
		MessageDigest complete = MessageDigest.getInstance("MD5");
		int numRead;
		do {
			numRead = fis.read(buffer);
			if (numRead > 0) {
				complete.update(buffer, 0, numRead);
				System.out.print(".");
			}
		} while (numRead != -1);
		fis.close();
		return complete.digest();
	}

	public static void main(String args[]) throws IOException {
		String file;
		String cfile;
		String ver = "SJEA 1.0";
		Scanner scanner = new Scanner(System.in);

		System.out

	.println("\n_____________________________________________________________")
;
		System.out
				.println("\n"
					+ ver
					+   "   VERIFY\nUsage:   java   Checksum
<decrypted-file> <checksum-file>\n");
		// prompt if no arguments
		if (args.length >= 1) {
			file = args[0];
		} else {
			System.out.print("\nEnter decrypted file: ");
			file = scanner.nextLine();
		}
		File decfile = new File(file);
		if (!decfile.exists()) {
			System.out.println("\nERROR!: Missing File: " + file);
			System.out

	.println("\n_____________________________________________________________")
;
			System.exit(1);
		}
		if (args.length >= 2) {
			cfile = args[1];
		} else {
			System.out.print("\nEnter checksum file: ");
			cfile = scanner.nextLine();
		}
		File chkfile = new File(cfile);
		if (!chkfile.exists()) {
			System.out.println("\nERROR!: Missing File: " + cfile);
			System.out

	.println("\n_____________________________________________________________")
;
			System.exit(1);
```

```
                }
                System.out.print("\n\n> Matching checksum: \"" + file + "\" & \""
                                + cfile + "\"...");
                check(file, cfile);
                System.out

        .println("\n_____________________________________________________________\n
");
        }
}

package sjea.statics;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Decryption.java" - decrypts a file
//  usage: java Decryption <input-file> <output-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.security.MessageDigest;
import java.util.Scanner;

public class Decryption {

        public static int create(String filename) {
                try {
                        // get checksum function
                        byte[] chk = createChecksum(filename);
                        String md5result = "";
                        for (int i = 0; i < chk.length; i++) {
                                // prints checksum in hex
                                md5result  +=  Integer.toString((chk[i]  &  0xff)  +
0x100, 16)
                                                .substring(1);
                        }
                        System.out.println("\n\nMD5: " + md5result);
                        return 1;
                } catch (Exception e) {
                        e.printStackTrace();
                        return 0;
                }
        }

        public static byte[] createChecksum(String filename) throws Exception {
                // create checksum (MessageDigest), write using buffer
                InputStream fis = new FileInputStream(filename);
                byte[] buffer = new byte[2000000];
                MessageDigest complete = MessageDigest.getInstance("MD5");
```

```
		int numRead;
		do {
			numRead = fis.read(buffer);
			if (numRead > 0) {
				complete.update(buffer, 0, numRead);
				System.out.print(".");
			}
		} while (numRead != -1);
		fis.close();
		return complete.digest();
	}

	public static void main(String args[]) throws IOException {
		FileInputStream in = null;
		FileOutputStream out = null;
		String file;
		String wfile;
		String ver = "SJEA 1.0";
		char[] pass;
		Scanner scanner = new Scanner(System.in);

		System.out

	.println("\n_____________________________________________________________")
;
		System.out
				.println("\n"
					+ ver
					+  "  DECRYPT\nUsage:  java  Decryption
<input-file> <output-file> <password>\n");
		if (args.length >= 1) {
			file = args[0];
		} else {
			System.out.print("\nEnter file to decrypt: ");
			file = scanner.nextLine();
		}
		File checkfile = new File(file);
		if (!checkfile.exists()) {
			System.out.println("\nERROR!: Missing File: " + file);
			System.out

	.println("\n_____________________________________________________________")
;
			System.exit(1);
		}
		if (args.length >= 2) {
			wfile = args[1];
		} else {
			System.out.print("\nEnter file to write to: ");
			wfile = scanner.nextLine();
		}
		if (args.length >= 3) {
			pass = args[2].toCharArray();
		} else {
			System.out.print("\nEnter password: ");
			pass = scanner.nextLine().toCharArray();
		}
		// set 16 byte vector array
		byte[] vector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte)
0x32,
				(byte) 0xF2, (byte) 0x2F, (byte) 0xF4, (byte) 0x6C,
				(byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03,
				(byte)  0xB4,  (byte)  0xA2,  (byte)  0xE7,  (byte)
```

```
0xC5, };
              // set in, out streams
              try {
                     in = new FileInputStream(file);
                     out = new FileOutputStream(wfile);
                     long len = checkfile.length();
                     System.out.print("\nInput file size: " + len + " bytes");
                     String pass_s = String.valueOf(pass);
                     byte[] pass_b = pass_s.getBytes();
                     String hcheck = "";
                     int c;
                     int p = 0;
                     int k = 15000;
                     int rp = 0;
                     int rv = 0;
                     System.out.print("\n\n> Decrypting.");
                     // continue while there is input
                     while ((c = in.read()) != -1) {
                            // p - header length - %ENC%
                            if (p > 4) {

                                   // write output
                                   out.write(c - vector[rv] - pass_b[rp]);
                                   // loop trough vector and password bytes
                                   rp = rp + 1;
                                   if (rp > pass_b.length - 1) {
                                          rp = 0;
                                   }
                                   rv = rv + 1;
                                   if (rv > vector.length - 1) {
                                          rv = 0;
                                   }

                            } else {
                                   String t_c = String.valueOf(c);
                                   hcheck = hcheck.concat(t_c);
                                   // check is file has header (is encrypted)
                                   if  (p  ==  4  &&  hcheck.equals("3769786737")  ==
false) {
                                          System.out.println("\n\nERROR!: File \""
+ file
                                                        + "\" is not encrypted!");
                                          System.out

       .println("\n_____________________________________________________________")
;
                                          System.exit(1);
                                   }
                            }
                            p = p + 1;
                            if (p == k) {
                                   System.out.print(".");
                                   k = k + 15000;
                            }
                     }
              } finally {
                     // close streams
                     in.close();
                     out.close();
              }
              // get the checksum of the file
              System.out.print("\n\n> Generating checksum...");
              create(wfile);
```

```
            File outfile = new File(wfile);
            long len = outfile.length();
            // print info
            System.out.println("\n\n\n\nDONE!\nOutput file \"" + wfile
                        + "\" created.\nFile size: " + len + " Bytes");
            System.out.println("\nPassword length: " + pass.length
                        + " character(s)");
            System.out

      .println("\n_____________________________________________________________\n
");
      }
}

package sjea.statics;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Encryption.java" - encrypts a file
//  usage: java Encryption <input-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.security.MessageDigest;
import java.util.Scanner;

public class Encryption {

      public static int create(String filename) {
            try {
                  // get checksum function
                  byte[] chk = createChecksum(filename);
                  File f = new File(filename + ".md5");
                  FileWriter of = new FileWriter(f);
                  String md5result = "";
                  for (int i = 0; i < chk.length; i++) {
                        // prints checksum in hex
                        md5result  +=  Integer.toString((chk[i]  &  0xff)  +
0x100, 16)
                                          .substring(1);
                  }
                  of.write(md5result);
                  of.close();
                  System.out.println("\n\nMD5: " + md5result);
                  return 1;
            } catch (Exception e) {
                  e.printStackTrace();
                  return 0;
```

```
		}
	}

	public static byte[] createChecksum(String filename) throws Exception {
		// create checksum (MessageDigest), write using buffer
		InputStream fis = new FileInputStream(filename);
		byte[] buffer = new byte[2000000];
		MessageDigest complete = MessageDigest.getInstance("MD5");
		int numRead;
		do {
			numRead = fis.read(buffer);
			if (numRead > 0) {
				complete.update(buffer, 0, numRead);
				System.out.print(".");
			}
		} while (numRead != -1);
		fis.close();
		return complete.digest();
	}

	public static void main(String args[]) throws IOException {
		FileInputStream in = null;
		FileOutputStream out = null;
		String file;
		String ver = "SJEA 1.0";
		char[] pass;
		Scanner scanner = new Scanner(System.in);

		System.out

	.println("\n_____________________________________________________________")
;
		System.out.println("\n" + ver
				+ "  ENCRYPT\nUsage:  java  Encryption  <input-file>
<password>\n");
		if (args.length >= 1) {
			file = args[0];
		} else {
			System.out.print("\nEnter file to encrypt: ");
			file = scanner.nextLine();
		}
		File checkfile = new File(file);
		if (!checkfile.exists()) {
			System.out.println("\nERROR!: Missing File: " + file);
			System.out

	.println("\n_____________________________________________________________")
;
			System.exit(1);
		}
		if (args.length >= 2) {
			pass = args[1].toCharArray();
		} else {
			System.out.print("\nEnter password: ");
			pass = scanner.nextLine().toCharArray();
		}
		// set 16 byte vector array
		byte[] vector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte)
0x32,
				(byte) 0xF2, (byte) 0x2F, (byte) 0xF4, (byte) 0x6C,
				(byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03,
				(byte) 0xB4, (byte) 0xA2, (byte) 0xE7, (byte)
0xC5, };
```

```
              // set in, out streams
              try {
                     in = new FileInputStream(file);
                     out = new FileOutputStream(file + ".enc");
                     long len = checkfile.length();
                     System.out.print("\nInput file size: " + len + " bytes");
                     // write header
                     byte[] hdr = "%ENC%".getBytes();
                     out.write(hdr);
                     String pass_s = String.valueOf(pass);
                     byte[] pass_b = pass_s.getBytes();
                     int c;
                     int p = 0;
                     int k = 15000;
                     int rv = 0;
                     int rp = 0;
                     System.out.print("\n\n> Encrypting...");

                     // continue while there is input
                     while ((c = in.read()) != -1) {
                            // write output
                            out.write(c + vector[rv] + pass_b[rp]);
                            // loop trough vector and password bytes
                            rp = rp + 1;
                            if (rp > pass_b.length - 1) {
                                   rp = 0;
                            }
                            rv = rv + 1;
                            if (rv > vector.length - 1) {
                                   rv = 0;
                            }
                     }
                     p = p + 1;
                     // k - print speed
                     if (p == k) {
                            System.out.print(".");
                            k = k + 15000;
                     }
              } finally {
                     // close streams
                     in.close(); out.close();          }
              // get the checksum of the file
              System.out.print("\n\n> Generating checksum...");
              create(file);
              File outfile = new File(file + ".enc");
              long len = outfile.length();
              // print info
              System.out.println("\n\n\n\nDONE!\nOutput file \"" + file
                            + ".enc\" created.\nFile size: " + len + " Bytes");
              System.out.println("\nChecksum   file   \""   +   file   +   ".md5\"
created.");
              System.out.println("\nPassword length: " + pass.length
                            + " character(s)");
       System.out.println("\n_____________________________________________________
________\n");

       }
}
```

Source Code A-1: SJEA Original Static Source Code

```
Package sjea;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Checksum.java" - checksum verification tool
//  usage: java Checksum <decrypted-file> <checksum-file>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.InputStream;
import java.security.MessageDigest;

public class Checksum {

	/**
	 * Pre-Refactoring: Common parameters are extracted as instance variable.
	 */
	private String file;

	/**
	 * Pre-Refactoring: Set of constructors.
	 */
	public Checksum() {
	}

	public Checksum(String file) {
		this.file = file;
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int check(String checksumFile) {
		int rc = 0;
		try {
			byte[] chk1 = createChecksum();
			BufferedReader	in	=	new	BufferedReader(new
FileReader(checksumFile));
			String str = "";
			str = in.readLine();
			in.close();
			String md5result1 = "";
			for (int i = 0; i < chk1.length; i++) {
				// prints checksum in hex
				md5result1	+=	Integer.toString((chk1[i]	&	0xff)	+
0x100, 16)
						.substring(1);
			}
			System.out.println("\n\n " + str + " < " + file);
			System.out.println("\n	"	+	md5result1	+	"	<	"	+
checksumFile);
```

```
                    // check if two checksums match
                    if (str.equals(md5result1)) {
                          System.out.println("\n\nDONE! Checksum match!");
                          rc = 1;
                    } else {
                          System.out.println("\n\nERROR!    Checksum    do    not
match!");
                          rc = 2;
                    }
                    in.close();
                    return rc;
              } catch (Exception e) {
                    e.printStackTrace();
                    return rc;
              }
       }

       /**
        * Pre-Refactoring: Common parameters are removed.
        */
       public byte[] createChecksum() throws Exception {
              // create checksum (MessageDigest), write using buffer
              InputStream fis = new FileInputStream(file);
              byte[] buffer = new byte[2000000];
              MessageDigest complete = MessageDigest.getInstance("MD5");
              int numRead;
              do {
                    numRead = fis.read(buffer);
                    if (numRead > 0) {
                          complete.update(buffer, 0, numRead);
                          System.out.print(".");
                    }
              } while (numRead != -1);
              fis.close();
              return complete.digest();
       }
}

package sjea;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Encryption.java" - encrypts a file
//  usage: java Encryption <input-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.InputStream;
import java.security.MessageDigest;

public class Encryption {
```

```
	/**
	 * Pre-Refactoring: Common parameters are extracted as instance variable.
	 */
	private String file;

	/**
	 * Pre-Refactoring: Set of constructors.
	 */
	public Encryption() {
	}

	public Encryption(String file) {
		this.file = file;
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int create() {
		try {
			// get checksum function
			byte[] chk = createChecksum();
			File f = new File(file + ".md5");
			FileWriter of = new FileWriter(f);
			String md5result = "";
			for (int i = 0; i < chk.length; i++) {
				// prints checksum in hex
				md5result  +=  Integer.toString((chk[i]  &  0xff)  +
0x100, 16)
					.substring(1);
			}
			of.write(md5result);
			of.close();
			System.out.println("\n\nMD5: " + md5result);
			return 1;
		} catch (Exception e) {
			e.printStackTrace();
			return 0;
		}
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public byte[] createChecksum() throws Exception {
		// create checksum (MessageDigest), write using buffer
		InputStream fis = new FileInputStream(file);
		byte[] buffer = new byte[2000000];
		MessageDigest complete = MessageDigest.getInstance("MD5");
		int numRead;
		do {
			numRead = fis.read(buffer);
			if (numRead > 0) {
				complete.update(buffer, 0, numRead);
				System.out.print(".");
			}
		} while (numRead != -1);
		fis.close();
		return complete.digest();
	}
}
```

```
package sjea;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Decryption.java" - decrypts a file
//  usage: java Decryption <input-file> <output-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.FileInputStream;
import java.io.InputStream;
import java.security.MessageDigest;

public class Decryption {

	/**
	 * Pre-Refactoring: Common parameters are extracted as instance variable.
	 */
	private String file;

	/**
	 * Pre-Refactoring: Set of constructors
	 */
	public Decryption() {
	}

	public Decryption(String file) {
		this.file = file;
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int create() {
		try {
			// get checksum function
			byte[] chk = createChecksum();
			String md5result = "";
			for (int i = 0; i < chk.length; i++) {
				// prints checksum in hex
				md5result  +=  Integer.toString((chk[i]  &  0xff)  +
0x100, 16)
                                        .substring(1);
			}
			System.out.println("\n\nMD5: " + md5result);
			return 1;
		} catch (Exception e) {
			e.printStackTrace();
			return 0;
		}
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
```

```
        */
       public byte[] createChecksum() throws Exception {
              // create checksum (MessageDigest), write using buffer
              InputStream fis = new FileInputStream(file);
              byte[] buffer = new byte[2000000];
              MessageDigest complete = MessageDigest.getInstance("MD5");
              int numRead;
              do {
                     numRead = fis.read(buffer);
                     if (numRead > 0) {
                            complete.update(buffer, 0, numRead);
                            System.out.print(".");
                     }
              } while (numRead != -1);
              fis.close();
              return complete.digest();
       }
}

package sjea;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Scanner;

public class SJEAApplication {

       public static void main(String args[]) throws IOException {
              // Encryption Application Function
              FileInputStream encin = null;
              FileOutputStream encout = null;
              String encfile;
              String encver = "SJEA 1.0";
              char[] encpass;
              Scanner scanner = new Scanner(System.in);

              System.out

       .println("\n_____________________________________________________________")
;
              System.out.println("\n" + encver
                            +  "  ENCRYPT\nUsage:  java  Encryption  <input-file>
<password>\n");
              if (args.length >= 1) {
                     encfile = args[0];
              } else {
                     System.out.print("\nEnter file to encrypt: ");
                     encfile = scanner.nextLine();
              }
              File checkfile = new File(encfile);
              if (!checkfile.exists()) {
                     System.out.println("\nERROR!: Missing File: " + encfile);
                     System.out

       .println("\n_____________________________________________________________")
;
                     System.exit(1);
              }
              if (args.length >= 2) {
                     encpass = args[1].toCharArray();
              } else {
```

```
                        System.out.print("\nEnter password: ");
                        encpass = scanner.nextLine().toCharArray();
                }
                // set 16 byte vector array
                byte[] vector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte)
0x32,
                                (byte) 0xF2, (byte) 0x2F, (byte) 0xF4, (byte) 0x6C,
                                (byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03,
                                (byte) 0xB4, (byte) 0xA2, (byte) 0xE7, (byte)
0xC5, };
                // set in, out streams
                try {
                        encin = new FileInputStream(encfile);
                        encout = new FileOutputStream(encfile + ".enc");
                        long len = checkfile.length();
                        System.out.print("\nInput file size: " + len + " bytes");
                        // write header
                        byte[] hdr = "%ENC%".getBytes();
                        encout.write(hdr);
                        String pass_s = String.valueOf(encpass);
                        byte[] pass_b = pass_s.getBytes();
                        int c;
                        int p = 0;
                        int k = 15000;
                        int rv = 0;
                        int rp = 0;
                        System.out.print("\n\n> Encrypting...");

                        // continue while there is input
                        while ((c = encin.read()) != -1) {
                                // write output
                                encout.write(c + vector[rv] + pass_b[rp]);
                                // loop trough vector and password bytes
                                rp = rp + 1;
                                if (rp > pass_b.length - 1) {
                                        rp = 0;
                                }
                                rv = rv + 1;
                                if (rv > vector.length - 1) {
                                        rv = 0;
                                }
                        }

                        p = p + 1;
                        // k - print speed
                        if (p == k) {
                                System.out.print(".");
                                k = k + 15000;
                        }
                } finally {
                        // close streams
                        encin.close();
                        encout.close();
                }
                // get the checksum of the file
                System.out.print("\n\n> Generating checksum...");
                Encryption enc = new Encryption(encfile);
                enc.create();
                File outfile = new File(encfile + ".enc");
                long len = outfile.length();
                // print info
                System.out.println("\n\n\n\nDONE!\nOutput file \"" + encfile
                                + ".enc\" created.\nFile size: " + len + " Bytes");
```

```
            System.out.println("\nChecksum file \"" + encfile + ".md5\"
created.");
            System.out.println("\nPassword length: " + encpass.length
                        + " character(s)");
            System.out

      .println("\n_____________________________________________________________\n
");

            // Decryption Application Function
            FileInputStream decin = null;
            FileOutputStream decout = null;
            String decfile;
            String decwfile;
            String decver = "SJEA 1.0";
            char[] decpass;
            Scanner decscanner = new Scanner(System.in);

            System.out

      .println("\n_____________________________________________________________")
;
            System.out
                        .println("\n"
                                    + decver
                                    + " DECRYPT\nUsage: java Decryption
<input-file> <output-file> <password>\n");
            if (args.length >= 1) {
                  decfile = args[0];
            } else {
                  System.out.print("\nEnter file to decrypt: ");
                  decfile = decscanner.nextLine();
            }
            File deccheckfile = new File(decfile);
            if (!deccheckfile.exists()) {
                  System.out.println("\nERROR!: Missing File: " + decfile);
                  System.out

      .println("\n_____________________________________________________________")
;
                  System.exit(1);
            }
            if (args.length >= 2) {
                  decwfile = args[1];
            } else {
                  System.out.print("\nEnter file to write to: ");
                  decwfile = decscanner.nextLine();
            }
            if (args.length >= 3) {
                  decpass = args[2].toCharArray();
            } else {
                  System.out.print("\nEnter password: ");
                  decpass = decscanner.nextLine().toCharArray();
            }
            // set 16 byte vector array
            byte[] decvector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8,
                        (byte) 0x32, (byte) 0xF2, (byte) 0x2F, (byte) 0xF4,
                        (byte) 0x6C, (byte) 0x56, (byte) 0x35, (byte) 0xE3,
                        (byte) 0x03, (byte) 0xB4, (byte) 0xA2, (byte) 0xE7,
                        (byte) 0xC5, };
            // set in, out streams
            try {
                  decin = new FileInputStream(decfile);
```

```
                    decout = new FileOutputStream(decwfile);
                    long declen = deccheckfile.length();
                    System.out.print("\nInput file size: " + declen + " bytes");
                    String pass_s = String.valueOf(decpass);
                    byte[] pass_b = pass_s.getBytes();
                    String hcheck = "";
                    int c;
                    int p = 0;
                    int k = 15000;
                    int rp = 0;
                    int rv = 0;
                    System.out.print("\n\n> Decrypting.");
                    // continue while there is input
                    while ((c = decin.read()) != -1) {
                           // p - header length - %ENC%
                           if (p > 4) {

                                  // write output
                                  decout.write(c - decvector[rv] - pass_b[rp]);
                                  // loop trough vector and password bytes
                                  rp = rp + 1;
                                  if (rp > pass_b.length - 1) {
                                         rp = 0;
                                  }
                                  rv = rv + 1;
                                  if (rv > decvector.length - 1) {
                                         rv = 0;
                                  }

                           } else {
                                  String t_c = String.valueOf(c);
                                  hcheck = hcheck.concat(t_c);
                                  // check is file has header (is encrypted)
                                  if (p == 4 && hcheck.equals("3769786737") ==
false) {
                                         System.out.println("\n\nERROR!: File \""
+ decfile
                                                       + "\" is not encrypted!");
                                         System.out

       .println("\n_____________________________________________________________")
;
                                         System.exit(1);
                                  }
                           }
                           p = p + 1;
                           if (p == k) {
                                  System.out.print(".");
                                  k = k + 15000;
                           }
                    }
             } finally {
                    // close streams
                    decin.close();
                    decout.close();
             }
             // get the checksum of the file
             System.out.print("\n\n> Generating checksum...");
             Decryption dec = new Decryption(decwfile);
             dec.create();
             File decoutfile = new File(decwfile);
             long declen = decoutfile.length();
             // print info
```

```
                System.out.println("\n\n\n\nDONE!\nOutput file \"" + decwfile
                            + "\" created.\nFile size: " + declen + " Bytes");
                System.out.println("\nPassword length: " + decpass.length
                            + " character(s)");
                System.out

        .println("\n_____________________________________________________________\n
");

                // Checksum Application Function
                String cksmfile;
                String cksmcfile;
                String cksmver = "SJEA 1.0";
                Scanner cksmscanner = new Scanner(System.in);

                System.out

        .println("\n_____________________________________________________________")
;
                System.out
                            .println("\n"
                                        + cksmver
                                        +   "   VERIFY\nUsage:   java   Checksum
<decrypted-file> <checksum-file>\n");
                // prompt if no arguments
                if (args.length >= 1) {
                        cksmfile = args[0];
                } else {
                        System.out.print("\nEnter decrypted file: ");
                        cksmfile = cksmscanner.nextLine();
                }
                File cksmdecfile = new File(cksmfile);
                if (!cksmdecfile.exists()) {
                        System.out.println("\nERROR!: Missing File: " + cksmfile);
        System.out.println("\n_____________________________________________________
________");
                        System.exit(1);
                }
                if (args.length >= 2) {
                        cksmcfile = args[1];
                } else {
                        System.out.print("\nEnter checksum file: ");
                        cksmcfile = cksmscanner.nextLine();
                }
                File chkfile = new File(cksmcfile);
                if (!chkfile.exists()) {
                        System.out.println("\nERROR!: Missing File: " + cksmcfile);
        System.out.println("\n_____________________________________________________
________"); System.exit(1);
                }
                System.out.print("\n\n> Matching checksum: \"" + cksmfile + "\" &
\""+ cksmcfile + "\"...");
                Checksum chksum = new Checksum(cksmfile);
                chksum.check(cksmcfile);
                System.out
        .println("\n_____________________________________________________________\n
");
        }
}
```

Source Code A-2: SJEA Original Domain Source Code

```
/**
 * 6.1. Extract Package: cryptography package.
 */

package sjearefactoring.flexibilityandextensibility.checksums;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Checksum.java" - checksum verification tool
//  usage: java Checksum <decrypted-file> <checksum-file>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.InputStream;
import java.security.MessageDigest;

public class Checksum {

	/**
	 * 1.4.1. Replace Magic Number with Symbolic Constant: MD5_ALGORITHM.
1.7.
	 * Push Down Field: MD5_ALGORITHM is pushed down to ChecksumMD5. 2.1.
Pull
	 * Up Field: MD5_ALGORITHM is pulled up to Checksum.
	 */
	public final static String MD5_ALGORITHM = "MD5";

	/**
	 * 1.4.1. Replace Magic Number with Symbolic Constant: CHECKSUM_SIZE.
	 */
	private final int CHECKSUM_SIZE = 2000000;

	/**
	 * Pre-Refactoring: Common parameters are extracted as instance variable.
	 */
	private String file;

	/**
	 * 1.6.2. Instance Variable: algorithm.
	 */
	private String algorithm;

	/**
	 * Pre-Refactoring: Set of constructors.
```

```
         */
        public Checksum() {
        }

        public Checksum(String file) {
                this.file = file;
        }

        /**
         * 1.3.2. Checksum constructor with parameter. 1.6.4. Add constructor
         * parameter.
         */
        public Checksum(String file, String algorithm) {
                this.file = file;
                this.algorithm = algorithm;
        }

        /**
         * 1.2. Encapsulate Field: getFile method. 1.3.1. Remove Setting Method:
         * setFile method is removed and a parameter is added to constructor.
         */
        public String getFile() {
                return file;
        }

        /**
         * 1.6.3. Encapsulate Field: getAlgorithm method.
         */
        public String getAlgorithm() {
                return algorithm;
        }

        /**
         * 1.6.3. Encapsulate Field: setAlgorithm.
         */
        public void setAlgorithm(String algorithm) {
                this.algorithm = algorithm;
        }

        /**
         * Pre-Refactoring: Common parameters are removed. 1.4.2. MD5_ALGORITHM
and
         * CHECKSUM_SIZE replace their values in createChecksum.
         */
        public byte[] createChecksum() throws Exception {
                // create checksum (MessageDigest), write using buffer
                InputStream fis = new FileInputStream(this.getFile());
                byte[] buffer = new byte[CHECKSUM_SIZE];
                MessageDigest                        complete                        =
MessageDigest.getInstance(this.getAlgorithm());
                int numRead;
                do {
                        numRead = fis.read(buffer);
                        if (numRead > 0) {
                                complete.update(buffer, 0, numRead);
                                writeToConsole(".");
                        }
                } while (numRead != -1);
                fis.close();
                return complete.digest();
        }

        /**
```

```
        * Pre-Refactoring: Common parameters are removed.
        */
       public int check(String checksumFile) {
              int rc = 0;
              try {
                     byte[] chk1 = this.createChecksum();
                     BufferedReader      in      =      new      BufferedReader(new
FileReader(checksumFile));
                     String str = "";
                     str = in.readLine();
                     in.close();
                     String md5result1 = "";
                     for (int i = 0; i < chk1.length; i++) {
                            // prints checksum in hex
                            md5result1  +=  Integer.toString((chk1[i]  &  0xff)  +
0x100, 16)
                                          .substring(1);
                     }
                     writeToConsole("\n\n " + str + " < " + this.getFile());
                     writeToConsole("\n " + md5result1 + " < " + checksumFile);
                     // check if two checksums match
                     if (str.equals(md5result1)) {
                            writeToConsole("\n\nDONE! Checksum match!");
                            rc = 1;
                     } else {
                            writeToConsole("\n\nERROR! Checksum do not match!");
                            rc = 2;
                     }
                     in.close();
                     return rc;
              } catch (Exception e) {
                     e.printStackTrace();
                     return rc;
              }
       }

       /**
        * 1.9. Extract Method: writeToConsole method.
        */
       protected void writeToConsole(String message) {
              System.out.println(message);
       }
}

/**
 * 6.1. Extract Package: cryptography package.
 */

package sjearefactoring.flexibilityandextensibility.checksums;

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

/**
 * 1.6.1. Extract Subclass: ChecksumMD5 is extracted and instance variable is
 * added to Checksum with getters and setters.
 */
public class ChecksumMD5 extends Checksum {

       /**
        * 1.6.5. ChecksumMD5 constructor with parameter invoking Checksum
```

```
        * constructor.
        */
       public ChecksumMD5(String file) {
              super(file, MD5_ALGORITHM);
       }

}

/**
 * 6.1. Extract Package: cryptography package.
 */

package sjearefactoring.flexibilityandextensibility.cryptography;

import sjearefactoring.flexibilityandextensibility.checksums.ChecksumMD5;

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

/**
 * 1.8.1. Extract Super-Class: Cryptography is extracted and common members of
 * Encryption and Decryption are pulled up. 2.3.1. Replace Delegation with
 * Inheritance: Cryptography extends ChecksumMD5.
 */
public class Cryptography extends ChecksumMD5 {

       /**
        * 1.1.2. Instance Variable: file. 1.8.3. Common instance variable is
pulled
        * up.
        */
       private String file;

       /**
        * 1.8.4. Cryptography constructor with parameter. 2.3.2. Cryptography
        * constructor invokes ChecksumMD5 constructor.
        */
       public Cryptography(String file) {
              super(file);
              this.file = file;
       }

       /**
        * 1.2. Encapsulate Field: getFile method. 1.3.1. Remove Setting Method:
        * setFile method is removed and a parameter is added to constructor.
1.8.3.
        * Common method is pulled up.
        */

       public String getFile() {
              return file;
       }

       /**
        * 1.9. Extract Method: writeToConsole method. 2.2. Pull Up Method:
        * writeToConsole method is pulled up to Cryptography.
        */
       protected void writeToConsole(String message) {
              System.out.println(message);
       }
```

```
	/**
	 * Pre-Refactoring: Common parameters are removed. 1.5. Hide Delegate:
	 * InputStream and MessageDigest are hidden by invoking createChecksum
	 * method of Checksum class. 2.2. Pull Up Method: createChecksum method
is
	 * pulled up to Cryptography. 2.3.3. createChecksum method is removed due
to
	 * replacing delegation to inheritance.
	 */
	/*
	 * public byte[] createChecksum() throws Exception { Checksum checksum =
new
	 * ChecksumMD5(this.getFile()); return checksum.createChecksum(); }
	 */
}

/**
 * 6.1. Extract Package: cryptography package.
 */

package sjearefactoring.flexibilityandextensibility.cryptography;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Decryption.java" - decrypts a file
//  usage: java Decryption <input-file> <output-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

/**
 * 1.8.2. Decryption extends Cryptography.
 */
public class Decryption extends Cryptography {

	/**
	 * 1.3.2. Decryption constructor with parameter. 1.8.5. Decryption
	 * constructor invokes Cryptography constructor.
	 */
	public Decryption(String file) {
		super(file);
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int create() {
		try {
			// get checksum function
			byte[] chk = createChecksum();
```

```
                    String md5result = "";
                    for (int i = 0; i < chk.length; i++) {
                          // prints checksum in hex
                          md5result  +=  Integer.toString((chk[i]  &  0xff)  +
0x100, 16)
                                      .substring(1);
                    }
                    writeToConsole("\n\nMD5: " + md5result);
                    return 1;
             } catch (Exception e) {
                    e.printStackTrace();
                    return 0;
             }
      }

      /**
       * Pre-Refactoring: Common parameters are removed. 1.5. Hide Delegate:
       * InputStream and MessageDigest are hidden by invoking createChecksum
       * method of Checksum class.
       */
      /*
       * public byte[] createChecksum() throws Exception { Checksum checksum =
new
       * ChecksumMD5(this.getFile()); return checksum.createChecksum(); }
       */
}

/**
 * 6.1. Extract Package: cryptography package.
 */

package sjearefactoring.flexibilityandextensibility.cryptography;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Encryption.java" - encrypts a file
//  usage: java Encryption <input-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

import java.io.File;
import java.io.FileWriter;

/**
 * 1.8.2. Encryption extends Cryptography.
 */
public class Encryption extends Cryptography {

      /**
```

```
        * 1.3.2. Encryption constructor with parameter. 1.8.5. Encryption
        * constructor invokes Cryptography constructor.
        */
       public Encryption(String file) {
              super(file);
       }

       /**
        * Pre-Refactoring: Common parameters are removed.
        */
       public int create() {
              try {
                     // get checksum function
                     byte[] chk = createChecksum();
                     File f = new File(this.getFile() + ".md5");
                     FileWriter of = new FileWriter(f);
                     String md5result = "";
                     for (int i = 0; i < chk.length; i++) {
                            // prints checksum in hex
                            md5result += Integer.toString((chk[i] & 0xff) +
0x100, 16)
                                          .substring(1);
                     }
                     of.write(md5result);
                     of.close();
                     writeToConsole("\n\nMD5: " + md5result);
                     return 1;
              } catch (Exception e) {
                     e.printStackTrace();
                     return 0;
              }
       }
}

package sjearefactoring.flexibilityandextensibility;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Scanner;

import sjearefactoring.flexibilityandextensibility.checksums.*;
import sjearefactoring.flexibilityandextensibility.cryptography.*;

public class SJEAApplication {

       public static void main(String args[]) throws IOException {
              // Encryption Application Function
              FileInputStream encin = null;
              FileOutputStream encout = null;
              String encfile;
              String encver = "SJEA 1.0";
              char[] encpass;
              Scanner scanner = new Scanner(System.in);

              System.out

       .println("\n_____________________________________________________________")
;
              System.out.println("\n" + encver
                           + " ENCRYPT\nUsage: java Encryption <input-file>
<password>\n");
```

```
              if (args.length >= 1) {
                     encfile = args[0];
              } else {
                     System.out.print("\nEnter file to encrypt: ");
                     encfile = scanner.nextLine();
              }
              File checkfile = new File(encfile);
              if (!checkfile.exists()) {
                     System.out.println("\nERROR!: Missing File: " + encfile);
                     System.out

       .println("\n_____________________________________________________________")
;
                     System.exit(1);
              }
              if (args.length >= 2) {
                     encpass = args[1].toCharArray();
              } else {
                     System.out.print("\nEnter password: ");
                     encpass = scanner.nextLine().toCharArray();
              }
              // set 16 byte vector array
              byte[] vector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte)
0x32,
                            (byte) 0xF2, (byte) 0x2F, (byte) 0xF4, (byte) 0x6C,
                            (byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03,
                            (byte) 0xB4, (byte) 0xA2, (byte) 0xE7, (byte)
0xC5, };
              // set in, out streams
              try {
                     encin = new FileInputStream(encfile);
                     encout = new FileOutputStream(encfile + ".enc");
                     long len = checkfile.length();
                     System.out.print("\nInput file size: " + len + " bytes");
                     // write header
                     byte[] hdr = "%ENC%".getBytes();
                     encout.write(hdr);
                     String pass_s = String.valueOf(encpass);
                     byte[] pass_b = pass_s.getBytes();
                     int c;
                     int p = 0;
                     int k = 15000;
                     int rv = 0;
                     int rp = 0;
                     System.out.print("\n\n> Encrypting...");

                     // continue while there is input
                     while ((c = encin.read()) != -1) {
                            // write output
                            encout.write(c + vector[rv] + pass_b[rp]);
                            // loop trough vector and password bytes
                            rp = rp + 1;
                            if (rp > pass_b.length - 1) {
                                   rp = 0;
                            }
                            rv = rv + 1;
                            if (rv > vector.length - 1) {
                                   rv = 0;
                            }
                     }

                     p = p + 1;
                     // k - print speed
```

```
                    if (p == k) {
                           System.out.print(".");
                           k = k + 15000;
                    }
             } finally {
                    // close streams
                    encin.close();
                    encout.close();
             }
             // get the checksum of the file
             System.out.print("\n\n> Generating checksum...");
             Encryption enc = new Encryption(encfile);
             enc.create();
             File outfile = new File(encfile + ".enc");
             long len = outfile.length();
             // print info
             System.out.println("\n\n\n\nDONE!\nOutput file \"" + encfile
                           + ".enc\" created.\nFile size: " + len + " Bytes");
             System.out.println("\nChecksum  file  \""  +  encfile  +  ".md5\"
created.");
             System.out.println("\nPassword length: " + encpass.length
                           + " character(s)");
             System.out

       .println("\n_____________________________________________________________\n
");

             // Decryption Application Function
             FileInputStream decin = null;
             FileOutputStream decout = null;
             String decfile;
             String decwfile;
             String decver = "SJEA 1.0";
             char[] decpass;
             Scanner decscanner = new Scanner(System.in);

             System.out

       .println("\n_____________________________________________________________")
;
             System.out
                           .println("\n"
                                        + decver
                                        +  "  DECRYPT\nUsage:  java  Decryption
<input-file> <output-file> <password>\n");
             if (args.length >= 1) {
                    decfile = args[0];
             } else {
                    System.out.print("\nEnter file to decrypt: ");
                    decfile = decscanner.nextLine();
             }
             File deccheckfile = new File(decfile);
             if (!deccheckfile.exists()) {
                    System.out.println("\nERROR!: Missing File: " + decfile);
                    System.out

       .println("\n_____________________________________________________________")
;
                    System.exit(1);
             }
             if (args.length >= 2) {
                    decwfile = args[1];
             } else {
```

```
                        System.out.print("\nEnter file to write to: ");
                        decwfile = decscanner.nextLine();
                }
                if (args.length >= 3) {
                        decpass = args[2].toCharArray();
                } else {
                        System.out.print("\nEnter password: ");
                        decpass = decscanner.nextLine().toCharArray();
                }
                // set 16 byte vector array
                byte[] decvector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8,
                                (byte) 0x32, (byte) 0xF2, (byte) 0x2F, (byte) 0xF4,
                                (byte) 0x6C, (byte) 0x56, (byte) 0x35, (byte) 0xE3,
                                (byte) 0x03, (byte) 0xB4, (byte) 0xA2, (byte) 0xE7,
                                (byte) 0xC5, };
                // set in, out streams
                try {
                        decin = new FileInputStream(decfile);
                        decout = new FileOutputStream(decwfile);
                        long declen = deccheckfile.length();
                        System.out.print("\nInput file size: " + declen + " bytes");
                        String pass_s = String.valueOf(decpass);
                        byte[] pass_b = pass_s.getBytes();
                        String hcheck = "";
                        int c;
                        int p = 0;
                        int k = 15000;
                        int rp = 0;
                        int rv = 0;
                        System.out.print("\n\n> Decrypting.");
                        // continue while there is input
                        while ((c = decin.read()) != -1) {
                                // p - header length - %ENC%
                                if (p > 4) {

                                        // write output
                                        decout.write(c - decvector[rv] - pass_b[rp]);
                                        // loop trough vector and password bytes
                                        rp = rp + 1;
                                        if (rp > pass_b.length - 1) {
                                                rp = 0;
                                        }
                                        rv = rv + 1;
                                        if (rv > decvector.length - 1) {
                                                rv = 0;
                                        }

                                } else {
                                        String t_c = String.valueOf(c);
                                        hcheck = hcheck.concat(t_c);
                                        // check is file has header (is encrypted)
                                        if (p == 4 && hcheck.equals("3769786737") ==
false) {
                                                System.out.println("\n\nERROR!: File \""
+ decfile
                                                                + "\" is not encrypted!");
                                                System.out

        .println("\n_____________________________________________________________")
;
                                                System.exit(1);
                                        }
                                }
```

```
                        p = p + 1;
                        if (p == k) {
                              System.out.print(".");
                              k = k + 15000;
                        }
                  }
            } finally {
                  // close streams
                  decin.close();
                  decout.close();
            }
            // get the checksum of the file
            System.out.print("\n\n> Generating checksum...");
            Decryption dec = new Decryption(decwfile);
            dec.create();
            File decoutfile = new File(decwfile);
            long declen = decoutfile.length();
            // print info
            System.out.println("\n\n\n\nDONE!\nOutput file \"" + decwfile
                        + "\" created.\nFile size: " + declen + " Bytes");
            System.out.println("\nPassword length: " + decpass.length
                        + " character(s)");
            System.out

      .println("\n_____________________________________________________________\n
");

            // Checksum Application Function
            String cksmfile;
            String cksmcfile;
            String cksmver = "SJEA 1.0";
            Scanner cksmscanner = new Scanner(System.in);

            System.out

      .println("\n_____________________________________________________________")
;
            System.out
                        .println("\n"
                                    + cksmver
                                    +   "   VERIFY\nUsage:   java   Checksum
<decrypted-file> <checksum-file>\n");
            // prompt if no arguments
            if (args.length >= 1) {
                  cksmfile = args[0];
            } else {
                  System.out.print("\nEnter decrypted file: ");
                  cksmfile = cksmscanner.nextLine();
            }
            File cksmdecfile = new File(cksmfile);
            if (!cksmdecfile.exists()) {
                  System.out.println("\nERROR!: Missing File: " + cksmfile);
                  System.out

      .println("\n_____________________________________________________________")
;
                  System.exit(1);
            }
            if (args.length >= 2) {
                  cksmcfile = args[1];
            } else {
                  System.out.print("\nEnter checksum file: ");
                  cksmcfile = cksmscanner.nextLine();
```

```
                }
                File chkfile = new File(cksmcfile);
                if (!chkfile.exists()) {
                        System.out.println("\nERROR!: Missing File: " + cksmcfile);
                        System.out

        .println("\n_____________________________________________________________")
;
                        System.exit(1);
                }
                System.out.print("\n\n> Matching checksum: \"" + cksmfile + "\" &
\""
                                + cksmcfile + "\"...");
                Checksum chksum = new Checksum(cksmfile);
                chksum.check(cksmcfile);
                System.out

        .println("\n_____________________________________________________________\n
");
        }
}
```

Source Code A-3: SJEA Refactored Source Code Using QARtF

```
/**
 * 2.1. Extract Package: checksums package.
 */
package sjealevelrefactoring.methodlevel.checksums;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Checksum.java" - checksum verification tool
//  usage: java Checksum <decrypted-file> <checksum-file>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.InputStream;
import java.security.MessageDigest;

public class Checksum {
	/**
	 * 3.1.1. Replace Magic Number with Symbolic Constant: CHECKSUM_SIZE.
	 */
	private final int CHECKSUM_SIZE = 2000000;

	/**
	 * 3.5. Pull Up Field: MD5_ALGORITHM is pulled up to Checksum.
	 */
	/**
	 * 3.1.1. Replace Magic Number with Symbolic Constant: MD5_ALGORITHM.
	 */
	public final static String MD5_ALGORITHM = "MD5";

	/**
	 * Pre-Refactoring: Common parameters are extracted as instance variable.
	 * 1.1.2. Instance Variable: algorithm.
	 */
	protected String file;
	protected String algorithm;

	/**
	 * Pre-Refactoring: Set of constructors.
	 */
	public Checksum() {
	}

	public Checksum(String file) {
		this.file = file;
	}

	/**
	 * 1.1.3. Add constructor parameter.
	 */
```

```
	public Checksum(String file, String algorithm) {
		this.file = file;
		this.algorithm = algorithm;
	}

	/**
	 * 3.2. Encapsulate Field: getAlgorithm method.
	 */
	public String getAlgorithm() {
		return algorithm;
	}

	/**
	 * 3.2. Encapsulate Field: setAlgorithm.
	 */
	public void setAlgorithm(String algorithm) {
		this.algorithm = algorithm;
	}

	/**
	 * 3.2. Encapsulate Field: getFile method.
	 */
	public String getFile() {
		return file;
	}

	/**
	 * 4.3. Remove Setting Method: setFile method.
	 */
	/**
	 * 3.2. Encapsulate Field: setFile.
	 */
	/*
	 * public void setFile(String file) { this.file = file; }
	 */

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int check(String checksumFile) {
		int rc = 0;
		try {
			byte[] chk1 = createChecksum();
			BufferedReader in = new BufferedReader(new
FileReader(checksumFile));
			String str = "";
			str = in.readLine();
			in.close();
			String md5result1 = "";
			for (int i = 0; i < chk1.length; i++) {
				// prints checksum in hex
				md5result1 += Integer.toString((chk1[i] & 0xff) +
0x100, 16)
						.substring(1);
			}
			writeToConsole("\n\n " + str + " < " + this.getFile());
			writeToConsole("\n " + md5result1 + " < " + checksumFile);
			// check if two checksums match
			if (str.equals(md5result1)) {
				writeToConsole("\n\nDONE! Checksum match!");
				rc = 1;
			} else {
				writeToConsole("\n\nERROR! Checksum do not match!");
```

```
                        rc = 2;
                }
                in.close();
                return rc;
            } catch (Exception e) {
                e.printStackTrace();
                return rc;
            }
        }

        /**
         * 4.1. Extract Method: writeToConsole method.
         */
        protected void writeToConsole(String message) {
            System.out.println(message);
        }

        /**
         *
         * Pre-Refactoring: Common parameters are removed.
         */
        /**
         * 3.1.2. CHECKSUM_SIZE replace its values in createChecksum method.
         */
        public byte[] createChecksum() throws Exception {
            // create checksum (MessageDigest), write using buffer
            InputStream fis = new FileInputStream(this.getFile());
            byte[] buffer = new byte[CHECKSUM_SIZE];
            MessageDigest complete = MessageDigest.getInstance("MD5");
            int numRead;
            do {
                numRead = fis.read(buffer);
                if (numRead > 0) {
                    complete.update(buffer, 0, numRead);
                    writeToConsole(".");
                }
            } while (numRead != -1);
            fis.close();
            return complete.digest();
        }
}

/**
 * 2.1. Extract Package: checksums package.
 */
package sjealevelrefactoring.methodlevel.checksums;

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

/**
 * 1.1.1 Extract Subclass: ChecksumMD5 is extracted and instance variable is
 * added to Checksum with getters and setters.
 */
public class ChecksumMD5 extends Checksum {
        /**
```

```
        * 3.5. Pull Up Field: MD5_ALGORITHM is pulled up to Checksum.
        */
       /**
        * 3.1.1. Replace Magic Number with Symbolic Constant: MD5_ALGORITHM.
        */
       /*
        * public final static String MD5_ALGORITHM = "MD5";
        */

       /**
        * 1.1.4. ChecksumMD5 constructor with parameter invoking Checksum
        * constructor.
        */
       /**
        * 3.1.2. MD5_ALGORITHM replace its values in ChecksumMD5 constructors.
        */
       public ChecksumMD5(String file) {
              super(file, MD5_ALGORITHM);
       }

}

/**
 * 2.1. Extract Package: cryptography package.
 */
package sjealevelrefactoring.methodlevel.cryptography;

/**
 * Refactoring SJEA to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */

import sjealevelrefactoring.methodlevel.checksums.ChecksumMD5;

/**
 * 1.2.1. Extract Super-Class: Cryptography is extracted and common members of
 * Encryption and Decryption are pulled up.
 */
/**
 * 1.5.1. Replace Delegation with Inheritance: Cryptography extends ChecksumMD5.
 */
public class Cryptography extends ChecksumMD5 {

       /**
        * 1.2.2. Instance Variable: file, algorithm.
        */
       protected String file;

       /**
        * 1.2.3. Cryptography constructors.
        */
       public Cryptography(String file) {
              super(file);
              this.file = file;
       }

       /**
        * 3.2. Encapsulate Field: getFile method.
        */
       public String getFile() {
              return file;
       }
```

```
	/**
	 * 4.3. Remove Setting Method: setFile method.
	 */
	/**
	 * 3.2. Encapsulate Field: setFile.
	 */
	/*
	 * public void setFile(String file) {
 this.file = file;
}
	 */

	/**
	 *  4.4.  Pull  Up  Method:  writeToConsole  method  is  pulled  up  to
Cryptography.
	 */
	protected void writeToConsole(String message) {
		System.out.println(message);
	}
}

/**
 * 2.1. Extract Package: cryptography package.
 */
package sjealevelrefactoring.methodlevel.cryptography;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Decryption.java" - decrypts a file
//  usage: java Decryption <input-file> <output-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.

/**
 * 1.1.3. Decryption extends Cryptography.
 */
public class Decryption extends Cryptography {

	/**
	 * 1.1.4. Decryption constructors.
	 */
	public Decryption(String file) {
		super(file);
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int create() {
		try {
			// get checksum function
			byte[] chk = createChecksum();
			String md5result = "";
			for (int i = 0; i < chk.length; i++) {
```

```
                        // prints checksum in hex
                        md5result += Integer.toString((chk[i] & 0xff) +
0x100, 16)
                                .substring(1);
                    }
                    writeToConsole("\n\nMD5: " + md5result);
                    return 1;
             } catch (Exception e) {
                    e.printStackTrace();
                    return 0;
             }
      }

      /**
       * 1.4. Hide Delegate: InputStream and MessageDigest are hidden by
invoking
       * createChecksum method of Checksum class.
       */
      /**
       * 1.5.2. createChecksum method is removed due to replacing delegation to
       * inheritance.
       */
      /*
       * public byte[] createChecksum(String filename) throws Exception
{ Checksum
       *       checksum       =       new       ChecksumMD5();       return
checksum.createChecksum(filename); }
       */

      /**
       * 4.4. Pull Up Method: writeToConsole method is pulled up to
Cryptography.
       */
      /**
       * 4.1. Extract Method: writeToConsole method.
       */
      /*
       * protected void writeToConsole(String message) {
       * System.out.println(message); }
       */
}

/**
 * 2.1. Extract Package: cryptography package.
 */
package sjealevelrefactoring.methodlevel.cryptography;

//  SJEA - Simple Java Encryption Algorithm 1.0
//  "Encryption.java" - encrypts a file
//  usage: java Encryption <input-file> <password>
//  Lubomir Ivanov - neolit123@gmail.com
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
import java.io.File;
import java.io.FileWriter;

/**
 * 1.1.3. Encryption extends Cryptography.
 */
public class Encryption extends Cryptography {

	/**
	 * 1.1.4. Encryption constructors.
	 */
	public Encryption(String file) {
		super(file);
	}

	/**
	 * Pre-Refactoring: Common parameters are removed.
	 */
	public int create() {
		try {
			// get checksum function
			byte[] chk = createChecksum();
			File f = new File(file + ".md5");
			FileWriter of = new FileWriter(f);
			String md5result = "";
			for (int i = 0; i < chk.length; i++) {
				// prints checksum in hex
				md5result += Integer.toString((chk[i] & 0xff) +
0x100, 16)
					.substring(1);
			}
			of.write(md5result);
			of.close();
			writeToConsole("\n\nMD5: " + md5result);
			return 1;
		} catch (Exception e) {
			e.printStackTrace();
			return 0;
		}
	}

	/**
	 * 4.4. Pull Up Method: writeToConsole method is pulled up to
Cryptography.
	 */
	/**
	 * 4.1. Extract Method: writeToConsole method.
	 */
	/*
	 * protected void writeToConsole(String message) {
	 * System.out.println(message); }
	 */

	/**
	 * 1.4. Hide Delegate: InputStream and MessageDigest are hidden by
invoking
	 * createChecksum method of Checksum class.
	 */
	/**
	 * 1.5.2. createChecksum method is removed due to replacing delegation to
	 * inheritance.
	 */
	/*
```

```
        * public byte[] createChecksum(String filename) throws Exception
{ Checksum
        *        checksum        =        new        ChecksumMD5();        return
checksum.createChecksum(filename); }
        */
}

package sjealevelrefactoring.methodlevel;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Scanner;

import sjealevelrefactoring.methodlevel.checksums.*;
import sjealevelrefactoring.methodlevel.cryptography.*;

public class SJEAApplication {

       public static void main(String args[]) throws IOException {
              // Encryption Application Function
              FileInputStream encin = null;
              FileOutputStream encout = null;
              String encfile;
              String encver = "SJEA 1.0";
              char[] encpass;
              Scanner scanner = new Scanner(System.in);

              System.out

       .println("\n_____________________________________________________________")
;
              System.out.println("\n" + encver
                           +  "  ENCRYPT\nUsage:  java  Encryption  <input-file>
<password>\n");
              if (args.length >= 1) {
                     encfile = args[0];
              } else {
                     System.out.print("\nEnter file to encrypt: ");
                     encfile = scanner.nextLine();
              }
              File checkfile = new File(encfile);
              if (!checkfile.exists()) {
                     System.out.println("\nERROR!: Missing File: " + encfile);
                     System.out

       .println("\n_____________________________________________________________")
;
                     System.exit(1);
              }
              if (args.length >= 2) {
                     encpass = args[1].toCharArray();
              } else {
                     System.out.print("\nEnter password: ");
                     encpass = scanner.nextLine().toCharArray();
              }
              // set 16 byte vector array
              byte[] vector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte)
0x32,
                           (byte) 0xF2, (byte) 0x2F, (byte) 0xF4, (byte) 0x6C,
                           (byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03,
                           (byte)  0xB4,  (byte)  0xA2,  (byte)  0xE7,  (byte)
```

```
0xC5, };
              // set in, out streams
              try {
                     encin = new FileInputStream(encfile);
                     encout = new FileOutputStream(encfile + ".enc");
                     long len = checkfile.length();
                     System.out.print("\nInput file size: " + len + " bytes");
                     // write header
                     byte[] hdr = "%ENC%".getBytes();
                     encout.write(hdr);
                     String pass_s = String.valueOf(encpass);
                     byte[] pass_b = pass_s.getBytes();
                     int c;
                     int p = 0;
                     int k = 15000;
                     int rv = 0;
                     int rp = 0;
                     System.out.print("\n\n> Encrypting...");

                     // continue while there is input
                     while ((c = encin.read()) != -1) {
                            // write output
                            encout.write(c + vector[rv] + pass_b[rp]);
                            // loop trough vector and password bytes
                            rp = rp + 1;
                            if (rp > pass_b.length - 1) {
                                   rp = 0;
                            }
                            rv = rv + 1;
                            if (rv > vector.length - 1) {
                                   rv = 0;
                            }
                     }

                     p = p + 1;
                     // k - print speed
                     if (p == k) {
                            System.out.print(".");
                            k = k + 15000;
                     }
              } finally {
                     // close streams
                     encin.close();
                     encout.close();
              }
              // get the checksum of the file
              System.out.print("\n\n> Generating checksum...");
              Encryption enc = new Encryption(encfile);
              enc.create();
              File outfile = new File(encfile + ".enc");
              long len = outfile.length();
              // print info
              System.out.println("\n\n\n\nDONE!\nOutput file \"" + encfile
                            + ".enc\" created.\nFile size: " + len + " Bytes");
              System.out.println("\nChecksum  file  \""  +  encfile  +  ".md5\"
created.");
              System.out.println("\nPassword length: " + encpass.length
                            + " character(s)");
              System.out

       .println("\n_____________________________________________________________\n
");
```

```
              // Decryption Application Function
              FileInputStream decin = null;
              FileOutputStream decout = null;
              String decfile;
              String decwfile;
              String decver = "SJEA 1.0";
              char[] decpass;
              Scanner decscanner = new Scanner(System.in);

              System.out

       .println("\n_____________________________________________________________")
;
              System.out
                           .println("\n"
                                        + decver
                                        +   "   DECRYPT\nUsage:   java   Decryption
<input-file> <output-file> <password>\n");
              if (args.length >= 1) {
                     decfile = args[0];
              } else {
                     System.out.print("\nEnter file to decrypt: ");
                     decfile = decscanner.nextLine();
              }
              File deccheckfile = new File(decfile);
              if (!deccheckfile.exists()) {
                     System.out.println("\nERROR!: Missing File: " + decfile);
                     System.out

       .println("\n_____________________________________________________________")
;
                     System.exit(1);
              }
              if (args.length >= 2) {
                     decwfile = args[1];
              } else {
                     System.out.print("\nEnter file to write to: ");
                     decwfile = decscanner.nextLine();
              }
              if (args.length >= 3) {
                     decpass = args[2].toCharArray();
              } else {
                     System.out.print("\nEnter password: ");
                     decpass = decscanner.nextLine().toCharArray();
              }
              // set 16 byte vector array
              byte[] decvector = { (byte) 0xA9, (byte) 0x9B, (byte) 0xC8,
                           (byte) 0x32, (byte) 0xF2, (byte) 0x2F, (byte) 0xF4,
                           (byte) 0x6C, (byte) 0x56, (byte) 0x35, (byte) 0xE3,
                           (byte) 0x03, (byte) 0xB4, (byte) 0xA2, (byte) 0xE7,
                           (byte) 0xC5, };
              // set in, out streams
              try {
                     decin = new FileInputStream(decfile);
                     decout = new FileOutputStream(decwfile);
                     long declen = deccheckfile.length();
                     System.out.print("\nInput file size: " + declen + " bytes");
                     String pass_s = String.valueOf(decpass);
                     byte[] pass_b = pass_s.getBytes();
                     String hcheck = "";
                     int c;
                     int p = 0;
                     int k = 15000;
```

```
                    int rp = 0;
                    int rv = 0;
                    System.out.print("\n\n> Decrypting.");
                    // continue while there is input
                    while ((c = decin.read()) != -1) {
                           // p - header length - %ENC%
                           if (p > 4) {

                                  // write output
                                  decout.write(c - decvector[rv] - pass_b[rp]);
                                  // loop trough vector and password bytes
                                  rp = rp + 1;
                                  if (rp > pass_b.length - 1) {
                                         rp = 0;
                                  }
                                  rv = rv + 1;
                                  if (rv > decvector.length - 1) {
                                         rv = 0;
                                  }

                           } else {
                                  String t_c = String.valueOf(c);
                                  hcheck = hcheck.concat(t_c);
                                  // check is file has header (is encrypted)
                                  if (p == 4 && hcheck.equals("3769786737") ==
false) {
                                         System.out.println("\n\nERROR!: File \""
+ decfile
                                                       + "\" is not encrypted!");
                                         System.out

       .println("\n_____________________________________________________________")
;
                                         System.exit(1);
                                  }
                           }
                           p = p + 1;
                           if (p == k) {
                                  System.out.print(".");
                                  k = k + 15000;
                           }
                    }
             } finally {
                    // close streams
                    decin.close();
                    decout.close();
             }
             // get the checksum of the file
             System.out.print("\n\n> Generating checksum...");
             Decryption dec = new Decryption(decwfile);
             dec.create();
             File decoutfile = new File(decwfile);
             long declen = decoutfile.length();
             // print info
             System.out.println("\n\n\n\nDONE!\nOutput file \"" + decwfile
                           + "\" created.\nFile size: " + declen + " Bytes");
             System.out.println("\nPassword length: " + decpass.length
                           + " character(s)");
             System.out

       .println("\n_____________________________________________________________\n
");
```

```
              // Checksum Application Function
              String cksmfile;
              String cksmcfile;
              String cksmver = "SJEA 1.0";
              Scanner cksmscanner = new Scanner(System.in);

              System.out

       .println("\n_____________________________________________________________")
;
              System.out
                           .println("\n"
                                        + cksmver
                                        +   "   VERIFY\nUsage:   java   Checksum
<decrypted-file> <checksum-file>\n");
              // prompt if no arguments
              if (args.length >= 1) {
                     cksmfile = args[0];
              } else {
                     System.out.print("\nEnter decrypted file: ");
                     cksmfile = cksmscanner.nextLine();
              }
              File cksmdecfile = new File(cksmfile);
              if (!cksmdecfile.exists()) {
                     System.out.println("\nERROR!: Missing File: " + cksmfile);
                     System.out

       .println("\n_____________________________________________________________")
;
                     System.exit(1);
              }
              if (args.length >= 2) {
                     cksmcfile = args[1];
              } else {
                     System.out.print("\nEnter checksum file: ");
                     cksmcfile = cksmscanner.nextLine();
              }
              File chkfile = new File(cksmcfile);
              if (!chkfile.exists()) {
                     System.out.println("\nERROR!: Missing File: " + cksmcfile);
                     System.out

       .println("\n_____________________________________________________________")
;
                     System.exit(1);
              }
              System.out.print("\n\n> Matching checksum: \"" + cksmfile + "\" &
\""
                           + cksmcfile + "\"...");
              Checksum chksum = new Checksum(cksmfile);
              chksum.check(cksmcfile);
              System.out

       .println("\n_____________________________________________________________\n
");
       }
}
```

Source Code A-4: SJEA Refactored Source Code Using LRtF

APPENDIX B: JFTP APPLICATION SOURCE CODE

Appendix B presents the source code of the JFTP application's classes before and after applying the refactoring to frameworks. It includes the original source code of the application and the resulted source codes after applying Quality Attribute Based Refactoring to Framework (QARtF) and Level-Based Refactoring to Framework (LRtF). Source Code B-5 contains JFTP Original Domain source code. Source Code B-6 contains JFTP refactored source code using QARtF. Source Code B-7 contains JFTP refactored source code using LRtF.

```
/*
 * Name:    Authenticateable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 17.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 06.06.2001 Tobias Kranz Changed name to __REAL__ english ;-)
 */
package jftp;

/**
 * Defines methods to authenticate a user.
 *
 * @since 0.0.2pA1
 */
public interface Authenticateable extends JFTPSuperInterface {
       /**
        * Authenticates an user.
        *
        * @return true if authentication succeds; otherwise false
        *@since 0.0.2pA1
        */
       public boolean authenticate();
}

/*
 * Name:    Changeable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 20.07.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Defines methods to change things.
 *
 * @since 0.1.99t4
 */
public interface Changeable extends JFTPSuperInterface {
       /**
        * Changes the remote working dir.
        *
        * @param A
        *            java.lang.String containing the new path.
        *@return true if succeeded; else false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean changeRemoteWorkingDir(String newPath);

       /**
        * Changes the local working dir.
        *
        * @param A
        *            java.lang.String containing the new path.
        *@return true if succeeded; else false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean changeLocalWorkingDir(String newPath);
```

```
       /**
        * Sets the Tx mode either to "ascii" or to "binary"
        *
        * @param A
        *            java.lang.String containing the wished Tx-mode (Either
        *            'binary' or 'ascii')
        *@return true if succeeded; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean setTxMode(String nm);

       /**
        * Sets the Tx mode either to "ascii" or to "binary"
        *
        * @param A
        *             char containing the wished Tx-mode (Either 'i'/'b' for
binary
        *            or 'a' for ascii)
        *@return true if succeeded; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean setTxMode(char nm);
}

/*
 * Name:    CommandLineParser.java (extends Parser (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation.
 * 0.0.2pA2 03.05.2001 Martin Loh   GET implemented
 *  0.0.2pA4  15.05.2001  Tobias  Kranz  fixed 'containsWildcards(String, int)'
&&
 *                                  'parse(String)'
 * 0.0.2pA4 05.06.2001 Tobias Kranz added 'license' command
 * 0.1.99   10.06.2001 Tobias Kranz added possibility to set Tx modes.
 *  0.1.99t2  17.07.2001  Tobias  Kranz  2nd  fix  of  'containsWildcards(String,
int)'
 *                                  added 'createPattern(String)' &&
 *                                            'comparePatternWithList(String,
String[])',
 *                                  implemented MGET and MPUT and the
 *                                  "local dir list" command-group.
 * 0.1.99t4 19.07.2001 Tobias Kranz added 'void pager(String)' && fixed bug
with
 *                                                    MGET  &&  added  'String
getKnownCommands()' &&
 *                                       implemented MKDIR && LMKDIR && PWD &&
LPWD
 * 0.1.99t5 23.07.2001 Tobias Kranz improved lcd
 */
package jftp;

import java.io.IOException;
import java.util.NoSuchElementException;
import java.util.StringTokenizer;
import java.util.Vector;

/**
```

```
 * Parses the CommandLine for known FTP-Commands.<BR>
 * This class is used by JFTP.<BR>
 *
 * @since 0.0.2pA1
 */
public class CommandLineParser extends Parser {
	private final char[] listLong1 = { 'l' };
	private final char[] listLong2 = { 'l', 's', ' ', '-', 'l' };
	private final char[] listShort1 = { 'l', 's' };
	private final char[] listShort2 = { 'd', 'i', 'r' };
	// the same for local
	private final char[] llistLong1 = { 'l', 'l' };
	private final char[] llistLong2 = { 'l', 'l', 's', ' ', '-', 'l' };
	private final char[] llistShort1 = { 'l', 'l', 's' };
	private final char[] llistShort2 = { 'l', 'd', 'i', 'r' };

	private FTPCmdServer fcs;
	private StdIn stdin = new StdIn();
	private StdOut stdout = new StdOut();
	private StdErr stderr = new StdErr();

	/**
	 * Constructor
	 *
	 * @since The beginning of Time.
	 */
	public CommandLineParser() {
		this.fcs = new FTPCmdServer();
	}

	/**
	 * Parses the CommandLine.
	 *
	 * @param s
	 *            String to parse.
	 *@version 6
	 *@since 0.0.2pA1
	 */
	public void parse(String s) {
		try {

			/*
			 * Have you ever wondered how the parsing works? There
are 3 main
			 * parts: At the first point we are checking for
commands that can
			 * be entered in any situation. At the second point we
are looking
			 * for commands that only makes sense if we're
connected. And at
			 * least there's a section at the end where all command
are handled
			 * which may entered in an unconnected status.
			 */

			s = s.trim();

			if (jftpSuper.getDebugLevel() >= 2)
				stderr.println("parsing \"" + s + "\"");

			// OPENs a connection
			if (s.startsWith("open")) {
				this.logon(s);
```

```
                    } else if (s.equals("info")) {
                          jftpSuper.printLongJFTPInfo();
                    } else if (s.equals("license")) {
                          pager(jftpSuper.getLicense());
                    } else if ((s.equals("quit")) || (s.equals("exit"))) {
                          stdout.println("obsolete. Use 'bye' instead.");
                    }
                    // LocalLiSt short
                    else if (s.equals(String.valueOf(llistShort1))
                                || s.equals(String.valueOf(llistShort2))) {
                          stdout.print(fcs.getLocalShortDirList());
                    }
                    // LocalLiSt Long
                    else if (s.equals(String.valueOf(llistLong1))
                                || s.equals(String.valueOf(llistLong2))) {
                          stdout.print(fcs.getLocalLongDirList());
                    }
                    // HELP
                    else if (s.equals("help")) {
                          String tmp;
                          tmp = getKnownCommands();

                          if (jftpSuper.isConnected()) {
                                tmp += "Server Commands:\n";
                                fcs.send(s); // Sends 'help' to the server
                                tmp += fcs.readCtrl(); // Reads the servers
commands.
                          }
                          pager(tmp);
                    }
                    // LMKDIR
                    else if (s.startsWith("lmkdir ")) {
                          if (fcs.createLocalDir(s.substring(7)))
                                stdout.println("Okay,   directory   \""   +
s.substring(7)
                                            + "\" was created.");
                          else
                                stdout.println("Unable    to    create    a
directory.");
                    }
                    // LRMDIR
                    else if (s.startsWith("lrmdir ")) {
                          if (fcs.removeLocalDir(s.substring(7)))
                                stdout.println("Okay,   directory   \""   +
s.substring(7)
                                            + "\" was removed.");
                          else
                                stdout.println("Unable   to   remove   that
directory.");
                    }
                    // LPWD
                    else if (s.equals("lpwd")) {
                          stdout.println(fcs.printLocalWorkingDir());
                    }
                    // LCD
                    else if (s.startsWith("lcd ")) {
                          if (!fcs.changeLocalWorkingDir(s.substring(4)))
                                stdout.println("Can't  change  working  dir
to: "
                                            + s.substring(4));
                          else
                                stdout.println("Okay,      working      dir
changed.");
```

```
                    }

                    // The following if's are only interresting if we're
connected
                    else if (jftpSuper.isConnected()) {
                           // CLOSEs a connection
                           if (s.equals("close")) {
                                  fcs.disconnect();
                           }
                           // Change working Directory
                           else if (s.startsWith("cd")) {

      fcs.changeRemoteWorkingDir(s.substring(3).trim());
                           }
                           // LiSt short
                           else if (s.equals(String.valueOf(listShort1))
                                         ||
s.equals(String.valueOf(listShort2))) {
                                  stdout.print(fcs.getShortDirList());
                           }
                           // LiSt Long
                           else if (s.equals(String.valueOf(listLong1))
                                         ||
s.equals(String.valueOf(listLong2))) {
                                  stdout.print(fcs.getLongDirList());
                           }
                           // GET
                           else if (s.startsWith("get")) {
                                  if (containsWildcards(s, 4)) {
                                         stdout
                                                .println("Wildcards are
NOT allowed. Use 'mget' instead.");
                                  } else {
                                         if
(!fcs.getFile(s.substring(4).trim(), s.substring(4)
                                                .trim()))
                                                stderr.println("Unable to get
the file.");
                                  }
                           }
                           // MGET
                           else if (s.startsWith("mget")) {
                                  String[]               tmp               =
comparePatternWithList(createPattern(s
                                                .substring(4).trim()),
fcs.getDirListArray());
                                  fcs.getFiles(tmp, tmp);
                           }
                           // PUT
                           else if (s.startsWith("put")) {
                                  if (containsWildcards(s, 4)) {
                                         stdout
                                                .println("Wildcards are
NOT allowed. Use 'mput' instead.");
                                  } else {
                                         if (!fcs.putFile(s.substring(4)))
                                                stdout.println("Unable to put
the file.");
                                  }
                           }
                           // MPUT
                           else if (s.startsWith("mput")) {
                                  String[]               tmp               =
```

```
comparePatternWithList(createPattern(s
                                        .substring(4).trim()),
fcs.getLocalDirListArray());
                                fcs.putFiles(tmp);
                            }
                            // setting the TX mode
                            else if (s.startsWith("type") || s.equals("bin")
                                        || s.equals("asc")) {
                                fcs.setTxMode(s);
                            }
                            // MKDIR
                            else if (s.startsWith("mkdir ")) {
                                fcs.createRemoteDir(s.substring(6));
                            }
                            // RMDIR
                            else if (s.startsWith("rmdir ")) {
                                fcs.removeRemoteDir(s.substring(6));
                            }
                            // PWD
                            /*
                             * Yes, this costs performance, but its easier to
change the
                             *  progx  behavior,  for  example  when  adapting
another protocol.
                             */
                            else if (s.equals("pwd")) {

       stdout.println(fcs.printRemoteWorkingDir());
                            } else if (s.equals("bye")) {
                                fcs.disconnect();
                                System.exit(0); // EXIT(0)
                            } else {
                                if  (s.length()  >  0)  // prevents  errors  if
you just type '\n'
                                {
                                    if (jftpSuper.getDebugLevel() >= 2)
                                        stdout.println("sending:  "  +
s);
                                    fcs.send(s);
                                    stdout.println(fcs.readCtrl());
                                }
                            }
                    } else // At this point we are not connected:
                    {
                            if (s.equals("bye")) {
                                System.exit(0); // EXIT(0)
                            } else {
                                if  (s.length()  >  0)  // prevents  errors  if
you just type '\n'
                                {
                                    stderr.println("Unable  to  execute  "
+ s + " .");
                                }
                            }
                    }
             } catch (StringIndexOutOfBoundsException siobe) {
                    if (jftpSuper.getDebugLevel() >= 2)
                            stderr
                                    .println("Error in main-loop of Clp:
"
                                                    + siobe.toString());
             } catch (NullPointerException npe) {
                    if (jftpSuper.getDebugLevel() >= 2)
```

```
                        stdout.println("NullPointerException caugth (in
clp)");
            } catch (NoSuchElementException nsee) {
                  if (jftpSuper.getDebugLevel() >= 2)
                        stdout.println("NoSuchElementException caugth (in
clp)");
            }
      }

      /**
       * Will be implemented when j2sdk1.4 is released!
       */
      /*
       * IS implemented! But in FTPCmdServer.\ private void getFiles(String
       * pattern) { }
       */

      /**
       * Manages the whole authentication/logon procedure
       *
       * @param s
       *            String containing all params entered
       *@since 0.0.2pA4
       *@version 2
       */
      private void logon(String s) {
            boolean serverIsValid = false;
            String server;

            /*
             * s.substring(5) will 'return false' if s.length < 5 !!!
             */
            if (s.length() > 5)
                  server = s.substring(5).trim();
            else
                  server = "";

            if (jftpSuper.getDebugLevel() >= 2)
                  stderr.println("server=\"" + server + "\"");

            if (jftpSuper.isConnected()) {
                  stdout
                              .println("Unable to re-connect; You must
disconnect first...");
            } else {
                  while (!serverIsValid) {
                        if (server.length() > 4)// What's the min length
of a valid inet
                        // Address?
                        {
                              serverIsValid = true;
                        } else {
                              stdout.print("(host): ");
                              try {
                                    server = stdin.readLine();
                              } catch (IOException ioe) {
                              }
                        }
                  }

                  if (jftpSuper.getDebugLevel() >= 2)
                        stdout.println("contacting server " + server);
```

```
                    if (jftpSuper.setServerTo(server)) // setting (new)
Server (Server
                    // is _really_ there...:~) )
                    {
                           fcs.connect(); // CONNECT !
                           if (jftpSuper.isConnected()) // connected ?
                           {
                                  // if (jftpSuper.getDebugLevel() >= 2)
                                  stdout.println(fcs.readCtrl());

                                  if (!fcs.authenticate()) {
                                         /*
                                          *    Should    be    handled    by
authenticate.
                                          */
                                         //     stderr.println("Unable     to
authenticate.");
                                         parse("close"); // close connection
from Server
                                  }
                           }
                    } else {
                           stderr.println("Cannot connect to server.");
                    }
             }
      }

      /**
       * Checks if the given String contains any wildcards ('*'/'?').
       *
       * @param s
       *            String to check
       *@param start
       *            int position to start at.
       *@return true if the given String contains any wildcards; otherwise
not
       *@since 0.0.2pA3
       */
      private boolean containsWildcards(String s, int start) {
             boolean rc = false;
             int i;

             // Check for a valid range
             if (start <= s.length()) {
                    for (i = start; i < s.length(); i++) {
                           if (((char) s.charAt(i) == '*') || ((char)
s.charAt(i) == '?')
                                         || ((char) s.charAt(i) == '[')) {
                                  rc = true;
                                  break;
                           }
                    }
             }

             return (rc);
      }

      /**
       * Creates a regular-expression-pattern usable by java.lang.util.regex
from
       * any posix-regex. Example: Posix-regex: java.lang.util.regex: (see
       * j2sdk1.4-doc) *[0-9]t.ab? {graph}*[0-9]t.ab{graph}? ||
       * {graph}*{num}?t.ab{graph}?
```

```
         *
         * @param pattern
         *            The posix-regex to compute
         *@return The matching java.lang.regex-regex
         *@since 0.1.99t2
         *@version 1
         */
        private String createPattern(String pattern) {
                // Initial things
                StringTokenizer st;
                String pat = pattern, // local-temporary pattern
                rc = "";

                if (jftpSuper.getDebugLevel() >= 2)
                        stderr.println("Pattern=" + pattern);
                /* <Parsing for '''s> */
                st = new StringTokenizer(pat, "*");

                if (st.hasMoreTokens())
                        rc += st.nextToken();

                while (st.hasMoreTokens())
                        rc += "{graph}*" + st.nextToken();

                // Check for start and end
                if (pat.charAt(0) == '*')
                        rc = "{graph}*" + rc;

                if ((pattern.charAt(pattern.length() - 1) == '*'))
                        rc += "{graph}*";
                /* </Parsing for '''s> */

                // Resetting
                st = null;
                pat = rc;
                rc = "";

                /* <Parsing for '?''s> */
                st = new StringTokenizer(pat, "?");

                if (st.hasMoreTokens())
                        rc += st.nextToken();

                while (st.hasMoreTokens())
                        rc += "{graph}?" + st.nextToken();

                // Check for start and end
                if (pat.charAt(0) == '?')
                        rc = "{graph}?" + rc;

                if ((pattern.charAt(pattern.length() - 1) == '?'))
                        rc += "{graph}?";
                /* </Parsing for '?''s> */

                /*
                 * Should be used before running another loop on the pattern.
//
                 * Resetting st = null; pat = rc; rc = "";
                 */
                return (rc);
        }

        /**
```

```
        * Compares a java.lang.String array with a given java.util.regex
compatible
        * regular expression and returns a list of all matching entries.
        *
        * @param pattern
        *            The java.util.regex pattern to use.
        *@param list
        *            A java.lang.String array which should be compared to the
        *            pattern
        *@return A java.lang.String array which all entries from list that
are
        *         matching the pattern.
        *@since 0.1.99t2
        *@version 1
        */
       private String[] comparePatternWithList(String pattern, String[] list)
{
              int i;
              java.util.regex.Pattern                    p                    =
java.util.regex.Pattern.compile(pattern);
              java.util.regex.Matcher m;
              Vector<String> tmp = new Vector<String>(1);

              for (i = 0; i < list.length; i++) {
                     m = p.matcher(list[i]);

                     if (m.matches())
                            tmp.add((String) list[i]);
              }

              // Vector tmp -> String rc[n]
              String[] rc = new String[tmp.size()];

              for (i = 0; i < tmp.size(); i++)
                     rc[i] = tmp.elementAt(i);

              return (rc);
       }

       /**
        * Show any txpe of text side by side. A pager. This method looks for
'\n''s
        * (24 times), then shows a "Press return.."-msg and waits for any
        * return-terminated input and restarts that loop until EOF.
        *
        * @param text
        *            The text to display
        *@since 0.1.99.test4
        *@version 3
        */
       private void pager(String text) {
              int i, j = 0;
              char c;

              text += "\n"; // Ugly way to make sure the prompt comes up in a
new
              // line.

              while (text.length() > j) {
                     for (i = 0; i < 23; i++) {
                            do {
                                   c = text.charAt(j++);
                                   stdout.print(c);
```

```
                        } while (c != '\n');
                    }

                    stdout.print("\nPress [Return] to continue...");
                    try {
                        stdin.readLine();
                    } catch (IOException whoCares) {
                    }
                }
        }

        /**
         * Returns a java.lang.String containing all commands known by 'this'.
         *
         * @return A java.lang.String containing all commands.
         *@since 0.1.99test4
         *@version 1
         */
        private String getKnownCommands() {
                return ("Commands recognized by JavaFTP: (* =>
unimplemented)\n"
                        + "\n"
                        + "Conntection-commands:\n"
                        + "\n"
                        + "open           Asks you the server's name or ip
to connect to.\n"
                        + "open SERVER    Opens a connection to the server
SERVER.\n"
                        + "close          Closes the active connection.\n"
                        + "\n"
                        + "List-commands:\n"
                        + "l\n"
                        + "ls -la           Shows a detailed list of the
active directory "
                        + "on the server.\n"
                        + "\n"
                        + "ls\n"
                        + "dir            Shows a short list of the active
directory on the server.\n"
                        + "\n"
                        + "ll\n"
                        + "lls -la          Shows a detailed list of the
active directory on "
                        + "the localhost.\n"
                        + "\n"
                        + "lls\n"
                        + "ldir           Shows a short list of the active
directory on "
                        + "the localhost.\n"
                        + "\n"
                        + "Options and switches:\n"
                        + "type X               Sets the Tx-mode to X.
(X=i(binary)||X=a(ascii))\n"
                        + "asc            Sets the Tx-mode to ascii.\n"
                        + "bin            Sets the Tx-mode to binary.\n"
                        + "\n"
                        + "Directory relative commands:\n"
                        + "cd PATH         Changes the active directory on
the server to PATH.\n"
                        + "lcd PATH        Changes the active directory on
the localhost to PATH.\n"
                        + "pwd                  Shows the current working
directory on the server.\n"
```

```
                            + "lpwd                Shows the current working
directory on the localhost.\n"
                            + "mkdir DIR      Creates a directory named DIR on
the server.\n"
                            + "rmdir DIR      Removes a directory named DIR on
the server.\n"
                            + "lmkdir DIR     Creates a directory named DIR on
the localhost.\n"
                            + "lrmdir DIR     Removes a directory named DIR on
the localhost.\n"
                            + "\n"
                            + "Transfer-commands:\n"
                            + "get FILE       Gets a file named FILE\n"
                            + "put FILE       Puts a file named FILE\n"
                            + "mget PATTERN   Gets all files that matches the
pattern PATTERN.\n"
                            + "mput PATTERN   Puts all files that matches the
pattern PATTERN.\n"
                            + "\n"
                            + "Misc stuff:\n"
                            + "info                Shows a short info about
JavaFTP.\n"
                            + "license        Shows the license under "
                            + "wich you are allowed to use JavaFTP.\n"
                            + "help           Shows this helpscreen.\n"
                            + "bye              Quits the program and closses
all open connections.\n"
                            + "\n");
      }
}

/*
 * Name:    Connectable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Defines methods to connect to a server.
 *
 * @since 0.0.2pA1
 */
public interface Connectable extends JFTPSuperInterface {
      public boolean connect();

      public boolean disconnect();

      /**
       * Opens a ctrl connection.
       *
       * @since 0.0.2pA4
       */
      public boolean openCtrlConnection();

      /**
       * Closes a ctrl connection.
       *
       * @since 0.0.2pA4
       */
```

```
        public boolean closeCtrlConnection();

        /**
         * Opens a passive data connection.
         *
         * @since 0.0.2pA4
         */
        public boolean openPassiveDataConnection();

        /**
         * Closes a passive data connection.
         *
         * @since 0.0.2pA4
         */
        public boolean closePassiveDataConnection();
}

/*
 * Name:    Createable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 19.07.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Defines methods to create directories and files.
 *
 * @since 0.1.99t4
 */
public interface Createable extends JFTPSuperInterface {
        /**
         * Creates a Directory on the server with the given name
         *
         * @return True if successfully created; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean createRemoteDir(String dirName);

        /**
         * Creates a Directory on the local fs with the given name
         *
         * @return True if successfully created; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean createLocalDir(String dirName);
}

/*
 * Name:    FTPCmdServer.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name               changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz       Creation
 * 0.0.2pA2 02.05.2001 Svenja Wittstadt   authenticate method implemented
 * 0.0.2pA3 06.05.2001 Tobias Kranz       added skeleton of 'int getPort()'.
 * 0.0.2pA4 15.05.2001 Tobias Kranz       changed 'String receiveLine()' &
 *                                        'String receiveBlock()'.
```

```
 * 0.0.2pA4 28.05.2001 Tobias Kranz       some small fixes...
 * 0.0.2pA4 06.06.2001 Tobias Kranz       REWRITE from scratch
 * 0.1.0    07.06.2001 Tobias Kranz &&
 *                     Sebastian Schipper Completed the rewrite.
 *                                        EVERYTHING´s FINE! ;-]
 * 0.1.0    09.06.2001 Martin Loh         Implemented the putFile Method and
 *                                        updated the getFile method
 * 0.1.99    10.06.2001 Tobias Kranz          Fixed bugs: cantListAfterPut,
cantGet.
 *                                        add 'setTxMode(String)' &
 *                                        'getFileSize(String)'
 * 0.1.99t1 11.06.2001 Tobias Kranz         added Progressbar in 'putFile()'
&&
 *                                        'getFile()'.
 * 0.1.99t1 12.06.2001 Tobias Kranz       added 'getFiles(String regex)'
 *                                            'not sure if this is the right
place..
 * 0.1.99t1 19.06.2001 Tobias Kranz       improved 'getFileSize(String)' I
 * 0.1.99t2 17.07.2001 Tobias Kranz       improved 'getFileSize(String)' II
 *                                        added 'getDirListArray()' &&
 *                                        'getLocalDirListArray()' &&
 *                                        'getLocalLongDirList()' &&
 *                                        'getLocalShortDirList()'
 * 0.1.99t4 19.07.2001 Tobias Kranz       Made an improvement suggestion in
 *                                                 'getFile()' && started to
implement
 *                                                   the "Createable" && the
"Removeable"
 *                                        interfaces.
 * 0.1.99t5 23.07.2001 Tobias Kranz        changed 'getFile()' && 'putFile()'
to
 *                                        work in the local working dir
 * 0.1.99t5 01.08.2001 Tobias Kranz       changed 'authenticate()'
 */
package jftp;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.BindException;
import java.net.ConnectException;
import java.net.MalformedURLException;
import java.net.NoRouteToHostException;
import java.net.ProtocolException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.net.UnknownServiceException;
import java.util.StringTokenizer;

/**
 * This class contains the implementation of all supported FTP-Commands
 *
 * @since 0.0.2pA1
 */
public final class FTPCmdServer extends JFTPSuper implements Connectable,
             Authenticateable,    Listable,    Transferable,    Createable,
Removeable,
```

```
            Changeable {
      private Socket ctrlSock;
      private Socket dataSock;

      private NetReader ctrlReader;
      private NetWriter ctrlWriter;
      private NetReader dataReader;
      private NetWriter dataWriter;

      private InputStream dataIs;
      private OutputStream dataOs;

      private InputStreamReader ctrlIsr;
      private InputStreamReader dataIsr;

      private StdIn stdin = new StdIn();
      private StdOut stdout = new StdOut();
      private StdErr stderr = new StdErr();

      /* <START Implementing "Changeable"> */
      public boolean changeRemoteWorkingDir(String newDir) {
            boolean rc = false;
            String receive;

            send("cwd " + newDir);
            receive = readCtrl();
            stdout.println(receive);

            if (receive.equals("250"))
                  rc = true;

            return (rc);
      }

      public boolean changeLocalWorkingDir(String newDir) {
            return (jftpSuper.setCurrentWorkingDir(newDir));
      }

      /**
       * Sets the transfermode either to ascii or to binary
       *
       * @param mode
       *            The mode to use ("ascii" or "binary")
       *@return true if mode is set, otherwise false
       *@since 0.1.99
       *@version 0
       */
      public boolean setTxMode(String mode) {
            boolean modeIsValid = false;

            if ((mode.equals("ascii")) || (mode.equals("asc"))) {
                  modeIsValid = true;
                  send("type a");
            } else if ((mode.equals("binary")) || (mode.equals("bin"))) {
                  modeIsValid = true;
                  send("type i");
            } else {
                  stderr.println("Unable to recognize the desired tx
mode.");
            }

            if (modeIsValid)
                  stdout.println(readCtrl());
```

```
		return (modeIsValid);
	}

	public boolean setTxMode(char nm) {
		boolean rc = false;

		if ((nm == 'i') || (nm == 'b'))
			rc = setTxMode("binary");
		else if (nm == 'a')
			rc = setTxMode("ascii");

		return (rc);
	}

	/* <END Implementing "Changeable"> */

	/* <START Implementing "Removeable"> */

	/**
	 * Removes a remote Directory called dirName
	 *
	 * @param dirName
	 *            The name of the Directory to remove
	 *@return true if successfully removed; otherwise false
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean removeRemoteDir(String dirName) {
		boolean rc = false; // senseless; but the compiler wants it ;-)
		String receive;

		send("rmd " + dirName.trim());
		receive = readCtrl();
		stdout.println(receive);

		if (receive.startsWith("250")) {
			rc = true;
		} else if (receive.startsWith("550")) {
			rc = false;
		}

		return (rc);
	}

	/**
	 * Removes a local Directory called dirName
	 *
	 * @param dirName
	 *            The name of the Directory to remove
	 *@return true if successfully removed; otherwise false
	 *@since 0.1.99t4
	 *@version 2
	 */
	public boolean removeLocalDir(String dirName) {
		String cd = jftpSuper.getCurrentWorkingDir();
		File rmDir = new File(cd.substring(0, cd.length() - 1) +
dirName.trim());

		return (rmDir.delete());
	}

	/* <END Implementing "Removeable"> */
```

```
	/* <START Implementing "Createable"> */

	/**
	 * Creates a directory on the server named dirName
	 *
	 * @param dirName
	 *            The name of the directory to create
	 *@return true if successfully created; otherwise false
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean createRemoteDir(String dirName) {
		boolean rc = false; // senseless; but the compiler wants it ;-)
		String receive;

		send("mkd " + dirName.trim());
		receive = readCtrl();
		stdout.println(receive);

		if (receive.startsWith("257")) {
			rc = true;
		} else if (receive.startsWith("521")) {
			rc = false;
		}

		return (rc);
	}

	/**
	 * Creates a directory on the localhost named dirName
	 *
	 * @param dirName
	 *            The name of the directory to create
	 *@return true if successfully created; otherwise false
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean createLocalDir(String dirName) {
		File newDir = new File(jftpSuper.getCurrentWorkingDir()
				+ File.separator + dirName.trim());

		return (newDir.mkdir());
	}

	/* <END Implementing "Createable"> */

	/* <START Implementing "Transferable"> */

	/**
	 * Reads data from the FTPServer.
	 *
	 * @return String
	 *@since 0.0.2pA4
	 *@version 3
	 */
	public String readCtrl() {
		String rc = "", line;
		int i = 0;

		try {
			do {
				line = "";
```

```
                        do {
                                i = getCtrlStream().read();
                                line += (char) i;
                        } while (i != 10);

                        rc += line;

                } while ((char) line.charAt(3) != ' ');
        } catch (Exception e) {
                stderr.println("Error  while  reading  from  server  (in
fcs): "
                                + e.toString());
        }

        return (rc);
    }

    /**
     * Reads a whole block of data from the ctrl-channel of the FTPServer.
     *
     * @return String
     *@since 0.0.2pA4
     *@version 2
     */
    public String readCtrlBlock() {
        String rc = "";
        int i = 0; // char read from server.

        try {
                do {
                        i = getCtrlStream().read();
                        if (i == -1)
                                break;
                        rc += (char) i;
                } while (true);
        } catch (IOException e) {
                stderr.println("Error receiving Block.");
        }

        return (rc);
    }

    /**
     * Reads data from the FTPServer.
     *
     * @return String
     *@since 0.0.2pA4
     *@version 1
     */
    public String readData() {
        String rc = "";
        int i = 0; // char read from server.

        try {
                i = getDataStream().read();

                while (i != -1) {
                        rc += (char) i;
                        i = getDataStream().read();
                        // stderr.println("retrieving: " + (char) i);
                }
                closePassiveDataConnection();
```

```
		} catch (IOException e) {
			stderr.println("Error receiving Block.");
		}

		return (rc);
	}

	/**
	 * Reads a whole block of data from the data-channel of the FTPServer.
	 *
	 * @return String
	 *@since 0.0.2pA4
	 *@version 0
	 */
	public String readDataBlock() {
		String rc = "";

		while ("this".equals("that")) {
			try {
				rc += getDataStream().read();
			} catch (Exception e) {
				stderr.println("Gotcha!");
			}
		} // YES, this does NOT work.

		return (rc);
	}

	/**
	 * Reads a single line of data from the data-channel of the FTPServer.
	 *
	 * @return String
	 *@since 0.0.2pA4
	 *@version 0
	 */
	public String readDataLine() {
		String rc = "";

		while ("this".equals("that")) {
			try {
				rc += getDataStream().read();
			} catch (Exception e) {
				stderr.println("Gotcha!");
			}
		} // YES, this does NOT work.

		return (rc);
	}

	/**
	 * Sends a String to the FTPServer.
	 *
	 * @param s
	 *            String to send
	 *@since 0.0.2pA1
	 */
	public void send(String s) {
		ctrlWriter.write(s);
	}

	/**
	 * Receives a list of files named 'localList' and tx's them to the
server.
```

```
	 *
	 * @param localList
	 *            a java.lang.String array where to save the files
	 *@return true if successfully transfered; else false
	 *@since 0.1.99t2
	 *@version 1
	 */
	public boolean putFiles(String[] localList) {
		boolean rc = true;
		int i;

		for (i = 0; i < localList.length; i++) {
			if (!putFile(localList[i]))
				rc = false;
		}

		return (rc);
	}

	/**
	 * Receives a list of files named 'serverList' and stores them in
	 * 'localList'
	 *
	 * @param serverList
	 *            a java.lang.String array of files to get
	 *@param localList
	 *            a java.lang.String array where to save the files
	 *@return true if successfully transfered; else false
	 *@since 0.1.99t2
	 *@version 1
	 */
	public boolean getFiles(String[] serverList, String[] localList) {
		boolean rc = true;
		int i;

		if (localList.length == serverList.length)
			for (i = 0; i < localList.length; i++) {
				if (!getFile(serverList[i], localList[i]))
					rc = false;
			}
		else
			rc = false;

		return (rc);
	}

	/**
	 * Recieves a File named 'ServerFile' and stores it at 'LocalFile'
	 *
	 * @param serverFile
	 *            The Filename at the Server's side
	 *@param localFile
	 *            The Filename where it is saved (local)
	 *@return True if the file is successfully transfered; else false
	 *@since 0.0.2pA1
	 *@version 2
	 */
	public boolean getFile(String serverFile, String localFile) {
		// We must catch errors like '5xx Permission denied...'
		boolean rc = false;
		int size = -1, i = 0;
		String serverResponse = "";
		FileOutputStream fos = null;
```

```
            size = getFileSize(serverFile);
            // stdout.println("Filesize is " + size);// Should only be used
for
            // debug(tk)

            if (openPassiveDataConnection()) {
                  if (size > 0) {
                        /*
                         * Here we may look first if the file already
exists on the
                         * local fs, compare the size and may only
transfer the missing
                         * part(s).(tk)
                         */
                        send("retr " + serverFile);
                        // stdout.println( "retrieving file" );//debug

                        // Checking return-value
                        serverResponse = readCtrl();
                        stdout.println(serverResponse);

                        if (serverResponse.startsWith("550")) {
                              stdout.println("Permission denied.");
                        } else if (serverResponse.startsWith("150")) { //
Server is
                              // ready to
                              // transfer.
                              try {
                                    Progressbar       pg       =       new
Progressbar(size);
                                    DataInputStream     dis     =     new
DataInputStream(
                                                getDataInputStream());
                                    i = 0;

                                    fos = new FileOutputStream(jftpSuper
                                                .getCurrentWorkingDir()
                                                + localFile);

                                    /*
                                     * This should be improved because
single byte reading
                                     * and writing is _VERY_ slow.
Maybe, there is a kind of
                                     * blockread and blockwrite
somewhere or we can connect
                                     * directly the dis and fos...?
                                     */
                                    /*    FML:    Whats    about    using
BufferedReader/Writer, too */
                                    while (i < size) {
                                          fos.write((byte) dis.read());
                                          i++;
                                          if ((i % 1024) == 0) // 'hope
this will save some
                                                // CPU-cycles
                                                pg.set(i);
                                    }
                                    pg.set(i); // set the Value again to
get 100% even on
                                    // small files.
```

```
                                        closePassiveDataConnection();
                                        fos.close();
                                        dis.close();
                                        rc = true;
                                  } catch (Exception e) {
                                        stderr.println("Error   while   Saving
File:"
                                                    + e.toString());
                                  }
                            }

                            // Closing socket && receiving server reply
                            try {
                                  stdout.println(readCtrl());
                                  if (jftpSuper.isPassiveConnected())
                                        closePassiveDataConnection();
                            } catch (Exception e) {
                                  stderr.println("Can't   Read   from   ctrl-
channel.");
                            }
                      }
                }

                return (rc);
          }

          /**
           * Puts a File 'localFile' to the Server
           *
           * @param localFile
           *            The file to transmit
           *@return  true  if  the  file  was  transfered  successfully;  otherwise
false
           *@since 0.0.2pA1
           *@version 5
           */
          public boolean putFile(String localFile) {
                // We must catch errors like '5xx Permission denied...'
                DataOutputStream dos;
                FileInputStream fis;
                int size, i;

                if (!openPassiveDataConnection()) {
                      stdout.println("Could    not    Open    Passive    Data
connection");
                      return (false);
                }
                send("stor " + localFile);
                stdout.println(readCtrl());

                try {
                      dos = new DataOutputStream(getDataOutputStream());
                      i = 0;
                      fis                          =                          new
FileInputStream(jftpSuper.getCurrentWorkingDir()
                                  + localFile);
                      /*
                       *  This  also  should  be  improved  because  single  byte
writing is
                       * _VERY_ slow.
                       */
                      size = fis.available();
```

```
                    Progressbar pb = new Progressbar(size);

                    while (i < size) {
                           dos.write((byte) fis.read());
                           i++;
                           if ((i % 1024) == 0)
                                  pb.set(i);
                    }
                    pb.set(i);

                    closePassiveDataConnection();
                    fis.close();
                    dos.close();
                    // FML: What about reading "270 ..." ?
                    stdout.println(readCtrl());
             } catch (Exception e) {
                    stdout
                                  .println("Exception occured while sending
data to the server: "
                                                + e.toString());
             }

             /* closePassiveDataConnection(); // FML: How often you want to
do this? */

             /* return ( false ); // FML: SURE???? */
             return (true);
       }

       /* <START Implementing tools for "Transferable"> */

       /* May someone improve this ugly code ? PLEASE */
       /**
        * Returns the size of a file named 'file'. It first tries to use the
size
        * command and falls back to a selfdetection mode by parsing the
output of a
        * 'list' command. If even this is not supported by the server it
fails.
        *
        * @param file
        *            The name of the file
        *@return The size of the given file; or -1 if it can't detect it.
        *@since 0.1.99
        *@version 3
        */
       private int getFileSize(String file) {
             /*
              * At first I'll try the size command, if it fails the method
tries to
              * determinate(?) the filesize by parsing the output of 'list'.
              */
             int rc = -1;
             String receive;

             send("size " + file);
             receive = readCtrl();

             if (!receive.startsWith("550")) {
                    try {
                           if (receive.startsWith("213") && receive.charAt(3)
== ' ')
                                  receive = receive.substring(3).trim();
```

```
                        rc = Integer.parseInt(receive);
                    } catch (Exception e) {
                        stderr.println("Server responses an invalid
filesize.");
                    }
                } else // Now me must go the hard way. :-(
                {
                    int i;
                    String list;

                    list = getLongDirList();

                    // Cutting the 1st and last line ("150 .." && "226 ..")
                    list = list.substring(list.indexOf("\n") + 1, // after
1st NL
                            list.lastIndexOf("\n", list.length() - 2)//
before last NL
                            );

                    /*
                     * Now we should have cut the first and last line. At
next we have
                     * to to replace '\n''s by ' ', so that the tokenizer
works proper.
                     * ARRG!, Then we have to replace '\r''s with ' ''s,
too.
                     */
                    list = list.replace('\n', ' ');
                    list = list.replace('\r', ' '); // Yes..

                    // Now we've got the String we want...going on.
                    StringTokenizer st = new StringTokenizer(list, " ");
                    String token;

                    boolean firstRun = true;
                    int size = 0;
                    int j = 1;

                    /*
                     * When tokenizing the first line, the needed fields are
8 & 4 and
                     * in any further run they are 9 & 5. Don't ask me
why... but tell
                     * me if you know it! ;-)
                     */
                    for (i = 0; st.hasMoreTokens(); i++) {
                        token = st.nextToken();

                        if (i == (9 - j)) {
                            i = 0;
                            if (firstRun) {
                                firstRun = false;
                                j = 0;
                            }
                            if (token.equals(file)) {
                                //  Since  '(5-j)'  is  smaller  than
'(9-j)' we can assign
                                // 'size' here.
                                rc = size;
                                break;
                            }
                        }
```

```
                        if (i == (5 - j))
                                size = Integer.parseInt(token);
                }
        }

        return (rc);
}

/**
 * Returns the current InputStreamReader of the ctrl-channel.
 *
 * @return InputStreamReader from the current ctrl-channel
 *@since 0.0.2pA4
 *@version 0
 */
private InputStreamReader getCtrlStream() {
        if (ctrlIsr == null) {
                this.ctrlIsr = ctrlReader.getStream();
        }

        return (ctrlIsr);
}

/**
 * Returns the current InputStreamReader of the data-channel.
 *
 * @return InputStreamReader from the current data-channel
 *@since 0.1.0
 *@version 0
 */
private InputStreamReader getDataStream() {
        if (dataIsr == null) {
                this.dataIsr = dataReader.getStream();
        }

        return (dataIsr);
}

/**
 * Returns the current InputStream of the data-channel.
 *
 * @return InputStream from the current data-channel
 *@since 0.1.99
 *@version 0
 */
private InputStream getDataInputStream() {
        if (dataIs == null) {
                this.dataIs = dataReader.getInputStream();
        }

        return (dataIs);
}

/**
 * Returns the current OutputStream of the data-channel.
 *
 * @return OutputStream from the current data-channel
 *@since 0.1.99
 *@version 0
 */
private OutputStream getDataOutputStream() {
        if (dataOs == null) {
                this.dataOs = dataWriter.getOutputStream();
```

```
		}

		return (dataOs);
	}

	/**
	 * Returns the port to use for PASV
	 *
	 * @param s
	 *            String to parse for the Port
	 *@return the port to use; or -1 if it can't resolve the port.
	 *@since 0.0.2pA1
	 */
	private int getPort(String s) {
		/*
		 * Now get the port to connect to.
		 *
		 * To do this we have to parse a string like this:
		 * "227 Entering Passive Mode (10,1,1,1,4,13)" where '10,1,1,1'
is the
		 * IP-Ad. and '4,13' stands for the port. ('4' are the first
8bits of an
		 * 16bit long number and '13' is the second.) In this case the
port
		 * would be 1037 (4256+13).
		 *
		 *   At   first   we   are   creating   a   substring   containing
"(10,1,1,1,4,13)" to
		 * parse and get the 5th and 6th Token, convert them to int and
		 * calculate the _REAL_ port to give it back.
		 */
		int rc = -1, port1, port2, i;
		StringTokenizer st;

		try {
			s = s.substring(s.indexOf('('), s.indexOf(')'));

			st = new StringTokenizer(s, ",", false);

			if (st.countTokens() == 6) {
				for (i = 0; i < 4; i++)
					st.nextToken();

				port1 = Integer.parseInt(st.nextToken());
				port2 = Integer.parseInt(st.nextToken());
				/*
				 * Now "calculate" them this way:
				 *
				 * 01010011 . 10101100 = 0101001110101100 ^ ^ ^ ^
8bit . 8bit =
				 * 16bit ^ ^ ^ ^ 1st port concatenation 2nd port =
_REAL_ port
				 */
				rc = (port1 * 256) + port2;
			}
		} catch (Exception e) {
			rc = -1;
		}

		return (rc);
	}

	/* <END Implementing tools for "Transferable"> */
```

```
	/* <END Implementing "Transferable"> */

	/* <START Implementing "Listable"> */

	/**
	 * Return the current working directory on the server.
	 *
	 * @return A java.lang.String containing the working directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printRemoteWorkingDir() {
		send("pwd");
		return (readCtrl());
	}

	/**
	 * Return the current working directory on the localhost.
	 *
	 * @return A java.lang.String containing the working directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printLocalWorkingDir() {
		return (jftpSuper.getCurrentWorkingDir());
	}

	/**
	 * Gives a detailed directorylist of the active directory.
	 *
	 * @return Directorylist of the active directory.
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public String getLongDirList() {
		String rc = "";

		if (openPassiveDataConnection()) {
			send("list");
			rc += readCtrl();
			if (!rc.startsWith("550")) {
				rc += readData();
				rc += readCtrl();
			}
		} else {
			rc = "Unable to establish a DataConnection.";
		}

		if (jftpSuper.isPassiveConnected())
			rc += "Unable to close the DataConnection.";

		return (rc);
	}

	/**
	 * Gives a short directorylist of the active directory.
	 *
	 * @return Directorylist of the active directory.
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public String getShortDirList() {
		String rc = "";
```

```
		boolean con;

		if (!jftpSuper.isPassiveConnected())
			con = openPassiveDataConnection();
		else
			con = true;

		if (con) {
			send("nlst");
			rc += readCtrl();
			if (!rc.startsWith("550")) {
				rc += readData();
				rc += readCtrl();
			} else {
				closePassiveDataConnection();
			}
		} else {
			rc = "Unable to establish a DataConnection.";
		}

		if (jftpSuper.isPassiveConnected())
			rc += "Unable to close the DataConnection.";

		return (rc);
	}

	/**
	 * Returns a java.lang.String array containing the directorylist of
the
	 * active directory.
	 *
	 * @return A java.lang.String array.
	 *@since 0.1.99t2
	 *@version 1
	 */
	public String[] getDirListArray() {
		int i;
		StringTokenizer st = new StringTokenizer(getShortDirList(),
"\n\r");
		String[] rc = new String[st.countTokens() - 2];
		String tmp = "";

		st.nextToken(); // pop "150 Openi..." to /dev/null

		for (i = 0; i < rc.length; i++) {
			tmp = st.nextToken();
			rc[i] = tmp;
		}

		st.nextToken(); // pop "226 Trans..." to /dev/null

		if (jftpSuper.getDebugLevel() >= 2)
			stderr.println("rc has # entries: " + rc.length);

		return (rc);
	}

	/**
	 * Returns a short directorylist of the current working-directory
	 *
	 * @return A java.lang.String
	 *@since 0.1.99t2
	 *@version 1
```

```
	 */
	public String getLocalShortDirList() {
		int i;
		String rc = "";
		File fi = new File(jftpSuper.getCurrentWorkingDir());
		String[] tmp = fi.list();

		for (i = 0; i < tmp.length; i++) {
			rc += tmp[i] + "\n\r";
		}

		return (rc);
	}

	/**
	 * Returns a long directorylist of the current working-directory
	 *
	 * @return A java.lang.String
	 *@since 0.1.99.t2
	 *@version 1
	 */
	public String getLocalLongDirList() {
		String rc = "";
		String[] tmp;
		File[] fl;
		int i;
		File fi = new File(jftpSuper.getCurrentWorkingDir());

		fl = fi.listFiles();
		tmp = new String[fl.length];

		for (i = 0; i < fl.length; i++) {
			// is Directory?
			if (fl[i].isDirectory())
				tmp[i] = "d";
			else
				tmp[i] = "-";

			// is Readable ?
			if (fl[i].canRead())
				tmp[i] += "r";
			else
				tmp[i] += "-";

			// is Writable ?
			if (fl[i].canWrite())
				tmp[i] += "w";
			else
				tmp[i] += "-";

			// Size
			// tmp[i] += " " + fl[i].length();

			tmp[i] += " " + fl[i].getName();
		}

		// String[] -> String
		for (i = 0; i < tmp.length; i++) {
			rc += tmp[i] + "\n\r";
		}

		return (rc);
	}
```

```
	/**
	 * Returns a java.lang.String arry containing the directorylist of the
local
	 * working-directory.
	 *
	 * @return A java.lang.String array.
	 *@since 0.1.99t2
	 *@version 1
	 */
	/*
	 * I know that this is not a really good place... ;-( (tk)
	 */
	public String[] getLocalDirListArray() {
		File fi = new File(jftpSuper.getCurrentWorkingDir());

		return (fi.list());
	}

	/* <END Implementing "Listable"> */

	/* <START Implementing "Authenticateable"> */

	/**
	 * Authenticates an user
	 *
	 * @return true if user is authenticated; otherwise false
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public boolean authenticate() {
		boolean rc = true;
		String passwd = "", strReceive;

		try {
			jftpSuper.unsetAuthenticated();
			stdout.print("Username: ");
			jftpSuper.setUserName(stdin.readLine()); // shouldn't we
check this?

			send("user " + jftpSuper.getUserName());
			strReceive = readCtrl();
			stdout.println(strReceive);
			if (strReceive.startsWith("331")) {
				if (jftpSuper.getUserName().equals("anonymous")) {
					passwd = jftpSuper.getAnonymousPasswd();
				} else {
					stdout.print("Password: ");
				}

				try {
					passwd = stdin.readPasswd();
				} catch (IOException ioe) {
					stderr.println("Unable    to    read    the
password.");
				}

				send("pass " + passwd);

				if (jftpSuper.getDebugLevel() >= 2)
					stderr.println("User:          "          +
jftpSuper.getUserName()
							+ " Pwd: " + passwd);
```

```
                        strReceive = this.readCtrl();
                        if (!(strReceive.startsWith("230"))) {
                              rc = false;
                        }
                        stdout.println(strReceive);
                  } else {
                        stderr.println("Login failed.");
                        rc = false;
                  }
            } catch (Exception e) {
                  rc = false;
                  //   stderr.println("Unable   to   authenticate.");//Only
debug(tk)
            }

            return (rc);
      }

      /* <END Implementing "Authenticateable"> */

      /* <START Implementing "Connectable"> */

      /**
       * Opens a connection
       *
       * @return true if connected; false if not
       *@since 0.1.0
       *@version 0
       */
      public boolean connect() {
            return (openCtrlConnection());
      }

      /**
       * Opens a ctrl connection
       *
       * @return true if connected; false if not
       *@since 0.1.0
       *@version 0
       */
      public boolean openCtrlConnection() {
            boolean rc = false;

            if ((!jftpSuper.isConnected()) && (bindCtrlSocket())) {
                  if (jftpSuper.getDebugLevel() >= 2)
                        stderr.println("Trying  to  get  Streams  from  the
ctrlSock.");

                  this.ctrlReader = new NetReader(ctrlSock);
                  this.ctrlWriter = new NetWriter(ctrlSock);

                  jftpSuper.setConnected();
                  rc = true;
            } else {
                  stderr.println("Unable to connect.");
            }

            return (rc);
      }

      /**
       * Closes a connection
```

```
         *
         * @return true if disconnected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean disconnect() {
               return (closeCtrlConnection());
        }

        /**
         * Closes a ctrl connection
         *
         * @return true if disconnected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean closeCtrlConnection() {
               boolean rc = false;

               if (jftpSuper.isConnected()) {
                      send("QuiT");
                      stdout.println(readCtrl());

                      try {
                             ctrlSock.close();
                             ctrlSock = null;
                             ctrlIsr = null;
                             ctrlReader = null;
                             ctrlWriter = null;

                             rc = true;
                      } catch (IOException ioe) {
                             System.out.println("Exception    while    closing
ControlSocket.");
                      }

                      jftpSuper.unsetConnected();
               }

               return (rc);
        }

        /**
         * Opens a passive connection for data.
         *
         * @return true if connected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean openPassiveDataConnection() {
               int port;
               boolean rc = true;
               String receive;

               if                ((jftpSuper.isConnected())                &&
(!jftpSuper.isPassiveConnected())) {
                      jftpSuper.unsetPassiveConnected();

                      send("pasv");
                      receive = readCtrl();
                      port = getPort(receive);

                      if (port == -1) {
```

```
                        rc = false;
                    } else {
                        if (bindDataSocket(port)) {
                            this.dataReader = new NetReader(dataSock);
                            this.dataWriter = new NetWriter(dataSock);

                            jftpSuper.setPassiveConnected();
                        } else // bindDataSocket(..) FAILED
                        {
                            rc = false;
                        }
                    }
            }

            return (rc);
        }

        /**
         * Closes a passive connection for data.
         *
         * @return true if disconnected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean closePassiveDataConnection() {
            boolean rc = false;

            if (jftpSuper.isPassiveConnected()) {
                try {
                    dataSock.close();
                    dataSock = null;
                    dataIs = null;
                    dataOs = null;
                    dataIsr = null;
                    dataReader = null;
                    dataWriter = null;

                    rc = true;
                } catch (IOException ioe) {
                    System.out.println("E...while             closeing
dataSock.");
                }
            }

            jftpSuper.unsetPassiveConnected();

            return (rc);
        }

        /* <START Implementing tools for "Connectable"> */
        /**
         * Binds a CtrlSocket to the Host/Port - pair set in JFTPSuper
         *
         * @return true if socket is bind; else false
         *@since 0.1.0
         *@version 0
         */
        private boolean bindCtrlSocket() {
            return (bindSocket('c', jftpSuper.getServerIP(), jftpSuper
                    .getServerPort()));
        }

        /**
```

```
        * Binds a DataSocket to the Host/Port - pair set in JFTPSuper
        *
        * @return true if socket is bind; else false.
        *@since 0.1.0
        *@version 0
        */
       private boolean bindDataSocket(int port) {
              return (bindSocket('d', jftpSuper.getServerIP(), port));
       }

       /**
        * Binds a Socket to the given Host/Port - pair
        *
        * @param sock
        *            The type of socket (only 'c' or 'd' are valid).
        *@param server
        *            The server to bind to.
        *@param port
        *            The Port to bind to.
        *@return true if socket is bind; else false
        *@since 0.1.0
        *@version 0
        */
       private boolean bindSocket(char sock, String server, int port) {
              boolean rc = false;

              if (jftpSuper.getDebugLevel() >= 2)
                     stderr.println("Binding Socket to server " + server +
":" + port);
              /* NO! ^ Not the beer */

              try {
                     if (sock == 'd')
                            dataSock = new Socket(server, port);
                     else if (sock == 'c')
                            ctrlSock = new Socket(server, port);
                     else
                            stderr.println("Ooops... You should __NEVER__ get
here! hum?");

                     rc = true;
              } catch (BindException be) {
                     stderr.println("Can't bind socket.");
              } catch (MalformedURLException mue) {
                     stderr.println("Malformed URL.");
              } catch (ConnectException ce) {
                     stderr.println("Connection refused by Server.");
              } catch (NoRouteToHostException nrthe) {
                     stderr.println("No route to host.");
              } catch (ProtocolException pe) {
                     stderr.println("A protocol error occured.");
              } catch (UnknownHostException uhe) {
                     stderr.println("Unable to resolve IP-address of
specified host.");
              } catch (UnknownServiceException use) {
                     stderr.println("Used protocol/service is not known.");
              } catch (IOException ioe) {
                     stderr.println("Unknown IO-Error.");
              }

              if (jftpSuper.getDebugLevel() >= 2)
                     stderr.println("Socket is bound (" + rc + ")");
```

```
                return (rc);
        }
        /* <END Implementing tools for "Connectable"> */
        /* <END Implementing "Connectable"> */
}

/*
 * Name:    JFTP.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name                      changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz              Creation
 * 0.0.2pA2 02.05.2001 Tobias Kranz              Added 'nextParamIsNumeric()'
 *                                               & 'nextParamExists()'
 * 0.0.2pA2 04.05.2001 Sebastian Schipper        Two bugs fixed
 */
package jftp;

import java.io.IOException;

/**
 * Main class.
 *
 * @since 0.0.2pA1
 */
public class JFTP extends JFTPSuper {
        private StdIn stdin;
        private StdOut stdout;
        private StdErr stderr;

        private final char[] guiShort = { '-', 'g' };
        private final char[] cliShort = { '-', 'c' };
        private final char[] helpShort = { '-', 'h' };
        private final char[] portShort = { '-', 'P' };
        private final char[] passiveShort = { '-', 'p' };
        private final char[] verboseShort = { '-', 'v' };
        private final char[] anonymousShort = { '-', 'a' };
        private final char[] aPasswdShort = { '-', 'A', 'p' };
        private final char[] guiLong = { '-', '-', 'g', 'u', 'i' };
        private final char[] cliLong = { '-', '-', 'c', 'l', 'i' };
        private final char[] helpLong = { '-', '-', 'h', 'e', 'l', 'p' };
        private final char[] portLong = { '-', '-', 'P', 'o', 'r', 't' };
        private final char[] verboseLong = { '-', '-', 'v', 'e', 'r', 'b',
'o',
                        's', 'e' };
        private final char[] passiveLong = { '-', '-', 'p', 'a', 's', 's',
'i',
                        'v', 'e' };
        private final char[] anonymousLong = { '-', '-', 'a', 'n', 'o', 'n',
'y',
                        'm', 'o', 'u', 's' };
        private final char[] aPasswdLong = { '-', '-', 'A', 'n', 'o', 'n',
'y',
                        'm', 'o', 'u', 's', 'P', 'a', 's', 's', 'w', 'd' };

        /**
         * Sets initial values.
         *
         * @since 0.0.2pA1
         */
        public JFTP() {
                this.stdin = new StdIn();
```

```
              this.stdout = new StdOut();
              this.stderr = new StdErr();
       }

       /**
        * Starts the prog.
        *
        * @param CommandLineArguments
        *                .
        *@since 0.0.2pA1
        */
       public static void main(String[] args) {
              JFTP j = new JFTP();
              j.parseArgs(args); // scanning cmdLine

              if (jftpSuper.isCli())
                     j.cli();
              else
                     j.gui();
       }

       /**
        * Starts the CommandLineInterface (cli) version.
        *
        * @since 0.0.2pA1
        */
       private void cli() {
              Parser clp = new CommandLineParser();

              if (jftpSuper.getDebugLevel() >= 1)
                     stdout.println("DebugLevel:               "              +
jftpSuper.getDebugLevel());
              stdout.println(jftpSuper.getJFTPInfo());
              stdout.println(jftpSuper.getCopyright());
              stdout.println(jftpSuper.getLicenseInfo());

              if (jftpSuper.isAutoconnect()) {
                     clp.parse("open " + jftpSuper.getServerIP());
              }

              while (true) {
                     stdout.print("jftp> ");

                     try {
                            clp.parse(stdin.readLine());
                     } catch (IOException e) {
                            stderr.println("Couldn't read from StdIn!");
                     }
              }
       }

       /**
        * Starts the GUI version.
        *
        * @since 0.0.2pA1
        */
       private void gui() {
              // GUI gui1 = new GUI();
       }

       /**
        * Parses the commandLine arguments.
        *
```

```
         * @param a
         *            String array to parse
         *@since 0.0.2pA1
         *@version 1
         */
        private void parseArgs(String[] a) {
                int i;

                try {
                        for (i = 0; i < a.length; i++) {
                                if (a[i].equals(String.valueOf(helpShort))
                                                ||
a[i].equals(String.valueOf(helpLong))) {
                                        this.showHelp();
                                } // Help
                                else if (a[i].equals(String.valueOf(passiveShort))
                                                ||
a[i].equals(String.valueOf(passiveLong))) {
                                        jftpSuper.setPassive();
                                } // Passive mode
                                else                                                    if
(a[i].equals(String.valueOf(anonymousShort))
                                                ||
a[i].equals(String.valueOf(anonymousLong))) {
                                        jftpSuper.setAnonymous();
                                } // Anonymous mode
                                else if (a[i].equals(String.valueOf(aPasswdShort))
                                                ||
a[i].equals(String.valueOf(aPasswdLong))) { // Anonymous
                                        // Passwd
                                        if (jftpSuper.isAnonymous()) {
                                                if (this.nextParamExists(a, i))

        jftpSuper.setAnonymousPasswdTo(a[++i]);
                                                else
                                                        this.showHelp();
                                        } else {
                                                stderr
                                                                .println("You  have  set
the anonymous password but not anonymous login!?!?");
                                                stderr.println("Exiting   with   error
(1).");
                                                System.exit(1);
                                        }

                                        if (jftpSuper.getDebugLevel() >= 2)
                                                stdout.println("Anonymous   Password:
"
                                                                                +
jftpSuper.getAnonymousPasswd());
                                } else if (a[i].equals(String.valueOf(guiShort))
                                                ||
a[i].equals(String.valueOf(guiLong))) {
                                        jftpSuper.setGui();
                                } // GUI
                                else if (a[i].equals(String.valueOf(cliShort))
                                                ||
a[i].equals(String.valueOf(cliLong))) {
                                        jftpSuper.setCli();
                                } // CLI
                                else if (a[i].equals(String.valueOf(verboseShort))
                                                ||
a[i].equals(String.valueOf(verboseLong))) { // Verbosity
```

```
                                    if       ((i       <       a.length)       &&
(this.nextParamIsNumeric(a, i))) {
                                          short     tmpDebug     =     (short)
Integer.parseInt(a[++i]);
                                          if ((tmpDebug <= 2) && (tmpDebug >=
0))

      jftpSuper.setDebugLevel(tmpDebug);
                                    } else {
                                          stderr.println("Missing    or    wrong
parameter.");
                                          this.showHelp();
                                    }
                              } // Port
                              else if (a[i].equals(String.valueOf(portShort))
                                          ||
a[i].equals(String.valueOf(portLong))) {
                                    if       ((i       <       a.length)       &&
(this.nextParamIsNumeric(a, i))) {
                                          int             tmpPort             =
Integer.parseInt(a[++i]);
                                          if  ((tmpPort  >=  1)  &&  (tmpPort  <=
65535)) // Ok !
                                          {

      jftpSuper.setServerPortTo(tmpPort);
                                                stdout.println("Don't  use  std
port. Using "
                                                            +
jftpSuper.getServerPort() + " instead.");
                                          } else {
                                                stdout

      .println("!WARNING! Specified port not in proper range.");
                                                stdout

      .println("!WARNING! Falling back to default port (21).");
                                          }
                                    } else {
                                          stderr.println("Missing    or    wrong
parameters");
                                          this.showHelp();
                                    }
                              } else {
                                    if (i == (a.length - 1)) // Last parameter
                                    {
                                          if (jftpSuper.setServerTo(a[i]))
                                                jftpSuper.setAutoconnect();
                                    } else {
                                          stdout.println("Unknown Option: '" +
a[i] + "'");
                                          this.showHelp();
                                    }
                              }
                        }

            } catch (Exception e) {
                  stdout.print("Exception: ");
                  e.printStackTrace();
            }

            //  if  (jftpSuper.isPassive())  stdout.println("PASSIVE  mode
prefered.");
```

```
		if (jftpSuper.isAnonymous())
			stdout.println("We'll logon as \"anonymous\".");
		// if (jftpSuper.isCli()) stdout.println("starting CLI.");
		if (jftpSuper.isGui())
			stdout.println("starting GUI.");
	}

	/**
	 * Checks if the next parameter is a numeric value.
	 *
	 * @param a
	 *            String array of parameters to check
	 *@param pos
	 *            current position in array a
	 *@return true if the next parameter exists and is numeric;otherwise
false
	 *@since 0.0.2pA2
	 */
	private boolean nextParamIsNumeric(String[] a, int pos) {
		boolean rc = false;
		int tmp;

		try {
			tmp = Integer.parseInt(a[(++pos)]);
			System.out.println(tmp);
			rc = true;
		} catch (IndexOutOfBoundsException iobe) {
			if (jftpSuper.getDebugLevel() >= 2)
				stderr.println("No more parameters.");
		} catch (NumberFormatException nfe) {
			if (jftpSuper.getDebugLevel() >= 2)
				stderr.println("No more parameters.");
		}

		return rc;
	}

	/**
	 * Checks if the next parameter exists.
	 *
	 * @param a
	 *            String array to check
	 *@param pos
	 *            current position in array a
	 *@return true if the next parameter exists; else false
	 *@since 0.0.2pA2
	 */
	private boolean nextParamExists(String[] a, int pos) {
		boolean rc = true;

		try {
			String tmp = a[++pos];
			System.out.println(tmp);
		} catch (IndexOutOfBoundsException iobe) {
			rc = false;
		}

		// maybe "(pos < a.size) ? (return true) : (return false)

		return rc;
	}

	/**
```

```
        * Shows the help screen.
        *
        * @since 0.0.1
        */
       private void showHelp() {
              /*
               * Verbosity level > 1 should only be used for development.
Stable
               * versions should only contain an option for setting verbosity
to true
               * or false.
               */
              stdout.println(jftpSuper.getJFTPInfo());
              stdout.println("usage: JFTP [-h[c|g]pa] [Server]");
              stdout
                            .println("          JFTP  [--help][--verbose  X][--
passive][--anonymous[--anonymousPasswd X]]");
              stdout.println("            [--port X][Server]\n");
              stdout.println("       -h,    --help                 Shows this
help");
              stdout
                            .println("                     -c,            --cli
Starts the CLI-version (Default).");
              stdout
                            .println("                     -g,            --gui
Starts the GUI-version.");
              stdout
                            .println("       -p,    --passive            Force
passive mode ftp.");
              stdout
                            .println("
(Default is active mode ftp(NOT now))");
              stdout
                            .println("                 -a,          --anonymous
Causes jftp to bypass normal login procedure");
              stdout
                            .println("                                       and
use anonymous login instead.");
              stdout
                            .println("            -Ap  X,  --AnonymousPasswd  X
Causes jftp to use X as anonymous password.");
              stdout
                            .println("       -v X,  --verbose X          Sets
the vorbosity level where X = 0 - 2.");
              stdout
                            .println("       -P X,  --Port X             Sets
the port number to X.");
              System.exit(0); // <- !EXIT!
       }
}

/*
 * Name:    JFTPIO.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Non usable class.
```

```
 *
 * @since 0.0.2pA1
 */
public class JFTPIO extends JFTPSuper {
}

/*
 * Name:    JFTPSuper.java
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name               changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz       Creation
 * 0.0.2pA2 30.04.2001 Tobias Kranz       added methods to handle variables/
 *                                        options
 *   0.0.2pA4  15.05.2001  Tobias  Kranz                added  option  for
'PassiveConnected'
 *  0.0.2pA4  28.05.2001  Tobias  Kranz              added  debuginfo  for
'setServerTo(..)'
 * 0.0.2pA4 05.06.2001 Tobias Kranz       added 'getLicense()' &
 *                                        'getCopyright()' method's
 * 0.0.2pA4 06.06.2001 Tobias Kranz       removed debuginfo for
 *                                        'setServerTo(..)'
 * 0.0.2pA5 07.06.2001 Sebastian Schipper fixed serious bug in status-
 *                                        methods
 * 0.1.99t4 20.07.2001 Tobias Kranz       added 'getCurrentWorkingDir()' &&
 *                                        'setCurrentWorkingDir(String)' &&
 *                                        'String getDirSeparator()'
 * 0.1.99t5 23.07.2001 Tobias Kranz       fixed 'setCurrentWorkingDir()' &&
 *                                        removed 'getDirSeparator()' (see
 *                                        JavaDoc:java.io.File for details)
 */
package jftp;

import java.io.File;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.BitSet;
import java.util.StringTokenizer;

/**
 * Parameter & Object class.
 *
 * @since 0.0.2pA1
 */
public class JFTPSuper {
       /**/
       protected static JFTPSuper jftpSuper = new JFTPSuper();
       /**/

       private final String buildDate = "01.08.2001";
       private final String version = "0.1.99test6";

       private BitSet options = new BitSet(5);
       private InetAddress server;
       private int port = 21;
       private String userName = "me";
       private String anonymousPasswd = "tux@northpole.org";
       private boolean authenticated = false;
       private File currentWorkingDir = new File(".");
       private short debugLevel = 0;
       private String info = "JavaFTP-Client v. " + version
                     + "                                                Build: " +
```

```
buildDate;

	public JFTPSuper() {
		/*
		 * BitSet 'options'
		 *
		 * #: if true: default: 0 => cli true 1 => passive true 2 =>
anonymous
		 * false 3 => connected false 4 => autoconnect false 5 =>
		 * passiveConnected false
		 */
		this.options.set(0);
		this.options.set(1);
		this.options.clear(2);
		this.options.clear(3);
		this.options.clear(4);
		this.options.clear(5);

		try {
			this.server = InetAddress.getByName("localhost");
		} catch (UnknownHostException uhe) {
			System.err.println("COUGHT!");
		}
	}

	/**
	 * Is Cli-mode running ?
	 *
	 * @return true if Cli-mode is running; false if not
	 *@since 0.0.2pA1
	 */
	protected boolean isCli() {
		return ((boolean) options.get(0));
	}

	/**
	 * Sets Cli-mode to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setCli() {
		options.set(0);
	}

	/**
	 * Sets Cli-mode to false.
	 *
	 * @since 0.0.2pA1
	 */
	protected void unsetCli() {
		options.clear(0);
	}

	/**
	 * Is Gui-mode running ?
	 *
	 * @return true if isCli() returns false and other way round.
	 *@since 0.0.2pA1
	 */
	protected boolean isGui() {
		return (((boolean) options.get(0) ? (false) : (true)));
	}
```

```
	/**
	 * Same than unsetCli.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setGui() {
		unsetCli();
	}

	/**
	 * Is passive connection mode prefered ?
	 *
	 * @return true if passive mode is prefered; false if not
	 *@since 0.0.2pA1
	 */
	protected boolean isPassive() {
		return ((boolean) options.get(1));
	}

	/**
	 * Sets passive mode to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setPassive() {
		options.set(1);
	}

	/**
	 * Sets passive mode to false.
	 *
	 * @since 0.0.2pA1
	 */
	protected void unsetPassive() {
		options.clear(1);
	}

	/**
	 * Is anonymous logon prefered ?
	 *
	 * @return true if anonymous logon is prefered; otherwise false
	 *@since 0.0.2pA1
	 */
	protected boolean isAnonymous() {
		return ((boolean) options.get(2));
	}

	/**
	 * Sets anonymous logon to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setAnonymous() {
		options.set(2);
	}

	/**
	 * Sets anonymous logon to false.
	 *
	 * @since 0.0.2pA1
	 */
	protected void unsetAnonymous() {
		options.clear(2);
```

```
	}

	/**
	 * Are we connected to a ftpserver ?
	 *
	 * @return true if connected; otherwise false
	 *@since 0.0.2pA1
	 */
	protected boolean isConnected() {
		return ((boolean) options.get(3));
	}

	/**
	 * Sets connected to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setConnected() {
		options.set(3);
	}

	/**
	 * Sets connected to false.
	 *
	 * @since 0.0.2pA1
	 */
	protected void unsetConnected() {
		options.clear(3);
	}

	/**
	 * Returns Autoconnect
	 *
	 * @return true if Autoconnect is true; otherwise false
	 *@since 0.0.2pA2
	 */
	protected boolean isAutoconnect() {
		return ((boolean) options.get(4));
	}

	/**
	 * Sets Autoconnect to true.
	 *
	 * @since 0.0.2pA2
	 */
	protected void setAutoconnect() {
		options.set(4);
	}

	/**
	 * Sets Autoconnect to false.
	 *
	 * @since 0.0.2pA2
	 */
	protected void unsetAutoconnect() {
		options.clear(4);
	}

	/**
	 * Are we passive Connected
	 *
	 * @return true if we are passive connected; otherwise false
	 *@since 0.0.2pA4
```

```
         *@version 0
         */
        protected boolean isPassiveConnected() {
               return (options.get(5));
        }

        /**
         * Sets passive connected to true
         *
         * @since 0.0.2pA4
         *@version 1
         */
        protected void setPassiveConnected() {
               options.set(5);
        }

        /**
         * Sets passive connected to false
         *
         * @since 0.0.2pA4
         *@version 1
         */
        protected void unsetPassiveConnected() {
               options.clear(5);
        }

        /**
         * Returns the current Server
         *
         * @return InetAddress of the Server
         *@since 0.0.2pA1
         */
        protected InetAddress getServer() {
               return (server);
        }

        /**
         * Returns the IP of the Server
         *
         * @return the servers IP
         *@since 0.0.2pA1
         */
        protected String getServerIP() {
               return (server.getHostAddress());
        }

        /**
         * Returns the Name of the Server
         *
         * @return the servers Name
         *@since 0.0.2pA1
         */
        protected String getServerName() {
               return (server.getHostName());
        }

        /**
         * sets the Servers IP/Name
         *
         * @param s
         *            Servers name/IP
         *@return true if the Server can be set; otherwise false
         *@version 2
```

```
         *@since 0.0.2pA1
         */
        protected boolean setServerTo(String s) {
               boolean rc;

               try {
                      server = InetAddress.getByName(s);

                      rc = true;
               } catch (UnknownHostException uhe) {
                      StdErr stderr = new StdErr();
                      stderr.println("Unable to reach host '" + s + "'.");
                      rc = false;
               }

               return (rc);
        }

        /**
         * What's the current port?
         *
         * @return The current port
         *@since 0.0.2pA1
         */
        protected int getServerPort() {
               return (port);
        }

        /**
         * Sets the port
         *
         * @param Port
         *            Port
         *@since 0.0.2pA1
         */
        protected void setServerPortTo(int Port) {
               port = Port;
        }

        /**
         * Returns the userName
         *
         * @return The current userName
         *@since 0.0.2pA2
         */
        protected String getUserName() {
               return (userName);
        }

        /**
         * Sets the userName
         *
         * @param s
         *            New userName
         *@since 0.0.2pA2
         */
        protected void setUserName(String s) {
               userName = s;
        }

        /**
         * Returns the anonymous passwd.
         *
```

```
         * @return The password used for anonymous login.
         *@since 0.0.2pA2
         */
        protected String getAnonymousPasswd() {
               return (anonymousPasswd);
        }

        /**
         * Sets the password used for anonymous login to s.
         *
         * @param s
         *            The password used for anonymous login.
         *@since 0.0.2pA2
         */
        protected void setAnonymousPasswdTo(String s) {
               anonymousPasswd = s;
        }

        /**
         * Are we authenticated ?
         *
         * @return true if we are authenticated; else false
         *@since 0.0.2pA2
         */
        protected boolean isAuthenticated() {
               return (authenticated);
        }

        /**
         * Sets authenticated to true
         *
         * @since 0.0.2pA2
         */
        protected void setAuthenticated() {
               authenticated = true;
        }

        /**
         * Sets authenticated to false
         *
         * @since 0.0.2pA2
         */
        protected void unsetAuthenticated() {
               authenticated = false;
        }

        /**
         * Returns a java.lang.String containing the current working directory
         *
         * @return A java.lang.String
         *@since 0.1.99t4
         *@version 1
         */
        protected String getCurrentWorkingDir() {
               String rc = currentWorkingDir.getAbsolutePath();
               rc = rc.substring(0, rc.length() - 1);

               return (rc);
        }

        /**
         * Sets the current working dir to the given java.lang.String
         *
```

```
        * @param A
        *            java.lang.String
        *@return true if path is set; otherwise false
        *@since 0.1.99t4
        *@version 2
        */
       /*
        * Some infos: Java's path-format is
        *  'DIR_SEPARATOR''PATH''DIR_SEPARATOR''.'.  eg:  /  PATH  /  .  .  eg:
"/root/."
        * || "C:\MICROS~1\."
        */
       protected boolean setCurrentWorkingDir(String newPath) {
              boolean rc = true;
              String cp;

              if (newPath.startsWith("/")) // absolute
              {
                     if (!(new File(newPath)).exists())
                            rc = false;
                     else
                            currentWorkingDir    =    new    File(newPath    +
File.separator + ".");
              } else if (newPath.startsWith("..")) // Dir up
              {
                     cp = getCurrentWorkingDir();
                     int  lastIndex  =  cp.length()  -  2;  // cutting  '\n'  from
the end
                     StringTokenizer   st   =   new   StringTokenizer(newPath,
File.separator);

                     do {
                            lastIndex = cp.lastIndexOf("/", lastIndex);
                            cp = cp.substring(0, lastIndex);
                            // pop ".." to /dev/null (we just need the 'st' as
counter)
                            st.nextToken();
                     } while ((st.hasMoreTokens()));

                     cp += File.separator + ".";

                     if (!(new File(cp)).exists())
                            rc = false;
                     else
                            currentWorkingDir = new File(cp);
              } else // relative
              {
                     cp = getCurrentWorkingDir();
                     if (cp.endsWith("."))
                            cp = cp.substring(0, cp.length() - 1);

                     if   (!(new   File(cp   +   newPath   +   File.separator   +
".")).exists())
                            rc = false;
                     else
                            currentWorkingDir  =  new  File(cp  +  newPath  +
File.separator
                                          + ".");
              }

              return (rc);
       }
```

```
	/**
	 * Returns the debugLevel
	 *
	 * @return The current debugLevel
	 *@since 0.0.2pA1
	 */
	protected short getDebugLevel() {
		return ((short) debugLevel);
	}

	/**
	 * Sets the debugLevel to the short-value of 's'
	 *
	 * @param s
	 *            New debugLevel
	 *@since 0.0.2pA1
	 */
	protected void setDebugLevel(short s) {
		debugLevel = s;
	}

	/**
	 * Returns a 80 character long info string.
	 *
	 * @return Info 'bout JFTP
	 */
	protected String getJFTPInfo() {
		return (info);
	}

	/**
	 * Returns a very long info string.
	 *
	 * @return Info 'bout JFTP
	 */
	protected String getLongJFTPInfo() {
		return ("\n"
				+ info
				+ "\n\n(c) 2000 - 2001 The JavaFTP-Group:\n\n"
				+ "    Uwe         Busch\n"
				+ "    Alexander   Flick\n"
				+ "    Christian   Kaminski\n"
				+ "    Tobias      Kranz       (TOBx@GMX.DE)\n"
				+ "    Martin      Loh         (MLoh80@GMX.DE)\n"
				+ "    Sebastian   Schipper    (dpi209@GMX.DE)\n"
				+ "    Mathias     Sroke\n"
				+   "              Dolapo                    Falola
(Dr_Steelface@HOTMAIL.COM)\n"
				+ "    Svenja      Wittstadt\n"
				+ "\n"
				+ "This software is distributed under the rules of
the General Public " + "License(GPL) version 2 or any later version.\n");
	}

	/**
	 * Returns the License valid for this application
	 *
	 * @return A String containing the used License
	 *@since 0.0.2pA4
	 *@version 1
	 */
	protected String getLicense() {
		return (" GNU GENERAL PUBLIC LICENSE\n Version 2, June 1991\n
```

\n Copyright (C) 1989, 1991 Free Software Foundation, Inc.\n 675 Mass Ave, Cambridge, MA 02139, USA\n Everyone is permitted to copy and distribute verbatim copies\n of this license document, but changing it is not allowed.\n \n Preamble\n \n The licenses for most software are designed to take away your\n freedom to share and change it. By contrast, the GNU General Public\n License is intended to guarantee your freedom to share and change free\n software--to make sure the software is free for all its users. This\n General Public License applies to most of the Free Software\n Foundation's software and to any other program whose authors commit to\n using it. (Some other Free Software Foundation software is covered by\n the GNU Library General Public License instead.) You can apply it to\n your programs, too.\n \n When we speak of free software, we are referring to freedom, not\n price. Our General Public Licenses are designed to make sure that you\n have the freedom to distribute copies of free software (and charge for\n this service if you wish), that you receive source code or can get it\n if you want it, that you can change the software or use pieces of it\n in new free programs; and that you know you can do these things.\n \n To protect your rights, we need to make restrictions that forbid\n anyone to deny you these rights or to ask you to surrender the rights.\n These restrictions translate to certain responsibilities for you if you\n distribute copies of the software, or if you modify it.\n \n For example, if you distribute copies of such a program, whether\n gratis or for a fee, you must give the recipients all the rights that\n you have. You must make sure that they, too, receive or can get the\n source code. And you must show them these terms so they know their\n rights.\n \n We protect your rights with two steps: (1) copyright the software, and\n (2) offer you this license which gives you legal permission to copy,\n distribute and/or modify the software.\n \n Also, for each author's protection and ours, we want to make certain\n that everyone understands that there is no warranty for this free\n software. If the software is modified by someone else and passed on, we\n want its recipients to know that what they have is not the original, so\n that any problems introduced by others will not reflect on the original\n authors' reputations.\n \n Finally, any free program is threatened constantly by software\n patents. We wish to avoid the danger that redistributors of a free\n program will individually obtain patent licenses, in effect making the\n program proprietary. To prevent this, we have made it clear that any\n patent must be licensed for everyone's free use or not licensed at all.\n \n The precise terms and conditions for copying, distribution and\n modification follow.\n \n GNU GENERAL PUBLIC LICENSE\n TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION\n \n 0. This License applies to any program or other work which contains\n a notice placed by the copyright holder saying it may be distributed\n under the terms of this General Public License. The \"Program\", below,\n refers to any such program or work, and a \"work based on the Program\"\n means either the Program or any derivative work under copyright law:\n that is to say, a work containing the Program or a portion of it,\n either verbatim or with modifications and/or translated into another\n language. (Hereinafter, translation is included without limitation in\n the term \"modification\".) Each licensee is addressed as \"you\".\n \n Activities other than copying, distribution and modification are not\n covered by this License; they are outside its scope. The act of\n running the Program is not restricted, and the output from the Program\n is covered only if its contents constitute a work based on the\n Program (independent of having been made by running the Program).\n Whether that is true depends on what the Program does.\n \n 1. You may copy and distribute verbatim copies of the Program's\n source code as you receive it, in any medium, provided that you\n conspicuously and appropriately publish on each copy an appropriate\n copyright notice and disclaimer of warranty; keep intact all the\n notices that refer to this License and to the absence of any warranty;\n and give any other recipients of the Program a copy of this License\n along with the Program.\n \n You may charge a fee for the physical act of transferring a copy, and\n you may at your option offer warranty protection in exchange for a fee.\n \n 2. You may modify your copy or copies of the Program or any portion\n of it, thus

forming a work based on the Program, and copy and\n distribute such modifications or work under the terms of Section 1\n above, provided that you also meet all of these conditions:\n \n a) You must cause the modified files to carry prominent notices\n stating that you changed the files and the date of any change.\n \n b) You must cause any work that you distribute or publish, that in\n \nwhole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third\n parties under the terms of this License.\n \n c) If the modified program normally reads commands interactively\n when run, you must cause it, when started running for such\n interactive use in the most ordinary way, to print or display an\n announcement including an appropriate copyright notice and a\n notice that there is no warranty (or else, saying that you provide\n a warranty) and that users may redistribute the program under\n these conditions, and telling the user how to view a copy of this\n License. (Exception: if the Program itself is interactive but\n does not normally print such an announcement, your work based on\n the Program is not required to print an announcement.)\n \n These requirements apply to the modified work as a whole. If\n identifiable sections of that work are not derived from the Program,\n and can be reasonably considered independent and separate works in\n themselves, then this License, and its terms, do not apply to those\n sections when you distribute them as separate works. But when you\n distribute the same sections as part of a whole which is a work based\n on the Program, the distribution of the whole must be on the terms of\n this License, whose permissions for other licensees extend to the\n entire whole, and thus to each and every part regardless of who wrote it.\n \n Thus, it is not the intent of this section to claim rights or contest\n your rights to work written entirely by you; rather, the intent is to\n exercise the right to control the distribution of derivative or\n collective works based on the Program.\n \n In addition, mere aggregation of another work not based on the Program\n with the Program (or with a work based on the Program) on a volume of\n a storage or distribution medium does not bring the other work under\n the scope of this License.\n \n 3. You may copy and distribute the Program (or a work based on it,\n under Section 2) in object code or executable form under the terms of\n Sections 1 and 2 above provided that you also do one of the following:\n \n a) Accompany it with the complete corresponding machine-readable\n source code, which must be distributed under the terms of Sections\n 1 and 2 above on a medium customarily used for software interchange; or,\n \n b) Accompany it with a written offer, valid for at least three\n years, to give any third party, for a charge no more than your\n cost of physically performing source distribution, a complete\n machine-readable copy of the corresponding source code, to be\n distributed under the terms of Sections 1 and 2 above on a medium\n customarily used for software interchange; or,\n \n c) Accompany it with the information you received as to the offer\n to distribute corresponding source code. (This alternative is\n allowed only for noncommercial distribution and only if you\n received the program in object code or executable form with such\n an offer, in accord with Subsection b above.)\n \n The source code for a work means the preferred form of the work for\n making modifications to it. For an executable work, complete source\n code means all the source code for all modules it contains, plus any\n associated interface definition files, plus the scripts used to\n control compilation and installation of the executable. However, as a\n special exception, the source code distributed need not include\n anything that is normally distributed (in either source or binary\n form) with the major components (compiler, kernel, and so on) of the\n operating system on which the executable runs, unless that component\n itself accompanies the executable.\n \n If distribution of executable or object code is made by offering\n access to copy from a designated place, then offering equivalent\n access to copy the source code from the same place counts as\n distribution of the source code, even though third parties are not\n compelled to copy the source along with the object code.\n \n 4. You may not copy, modify, sublicense, or distribute the Program\n except as expressly provided under this License. Any attempt\n otherwise to copy, modify,

sublicense or distribute the Program is\n void, and will automatically terminate your rights under this License.\n However, parties who have received copies, or rights, from you under\n this License will not have their licenses terminated so long as such\n parties remain in full compliance.\n \n 5. You are not required to accept this License, since you have not\n signed it. However, nothing else grants you permission to modify or\n distribute the Program or its derivative works. These actions are\n prohibited by law if you do not accept this License. Therefore, by\n modifying or distributing the Program (or any work based on the\n Program), you indicate your acceptance of this License to do so, and\n all its terms and conditions for copying, distributing or modifying\n the Program or works based on it.\n \n 6. Each time you redistribute the Program (or any work based on the\n Program), the recipient automatically receives a license from the\n original licensor to copy, distribute or modify the Program subject to\n these terms and conditions. You may not impose any further\n restrictions on the recipients' exercise of the rights granted herein.\n You are not responsible for enforcing compliance by third parties to\n \nthis License. \n \n7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues),\n conditions are imposed on you (whether by court order, agreement or\n otherwise) that contradict the conditions of this License, they do not\n excuse you from the conditions of this License. If you cannot\n distribute so as to satisfy simultaneously your obligations under this\n License and any other pertinent obligations, then as a consequence you\n may not distribute the Program at all. For example, if a patent\n license would not permit royalty-free redistribution of the Program by\n all those who receive copies directly or indirectly through you, then\n the only way you could satisfy both it and this License would be to\n refrain entirely from distribution of the Program.\n \n If any portion of this section is held invalid or unenforceable under\n any particular circumstance, the balance of the section is intended to\n apply and the section as a whole is intended to apply in other\n circumstances.\n \n It is not the purpose of this section to induce you to infringe any\n patents or other property right claims or to contest validity of any\n such claims; this section has the sole purpose of protecting the\n integrity of the free software distribution system, which is\n implemented by public license practices. Many people have made\n generous contributions to the wide range of software distributed\n through that system in reliance on consistent application of that\n system; it is up to the author/donor to decide if he or she is willing\n to distribute software through any other system and a licensee cannot\n impose that choice.\n \n This section is intended to make thoroughly clear what is believed to\n be a consequence of the rest of this License.\n \n 8. If the distribution and/or use of the Program is restricted in\n certain countries either by patents or by copyrighted interfaces, the\n original copyright holder who places the Program under this License\n may add an explicit geographical distribution limitation excluding\n those countries, so that distribution is permitted only in or among\n countries not thus excluded. In such case, this License incorporates\n the limitation as if written in the body of this License.\n \n 9. The Free Software Foundation may publish revised and/or new versions\n of the General Public License from time to time. Such new versions will\n be similar in spirit to the present version, but may differ in detail to\n address new problems or concerns.\n \n Each version is given a distinguishing version number. If the Program\n specifies a version number of this License which applies to it and \"any\n later version\", you have the option of following the terms and conditions\n either of that version or of any later version published by the Free\n Software Foundation. If the Program does not specify a version number of\n this License, you may choose any version ever published by the Free Software\n Foundation.\n \n 10. If you wish to incorporate parts of the Program into other free\n programs whose distribution conditions are different, write to the author\n to ask for permission. For software which is copyrighted by the Free\n Software Foundation, write to the Free Software Foundation; we sometimes\n make exceptions for this. Our decision will be

```
guided by the two goals\n of preserving the free status of all derivatives
of our free software and\n of promoting the sharing and reuse of software
generally.\n \n NO WARRANTY\n \n 11. BECAUSE THE PROGRAM IS LICENSED FREE OF
CHARGE, THERE IS NO WARRANTY\n FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN\n OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES\n PROVIDE THE PROGRAM \"AS IS\" WITHOUT
WARRANTY OF ANY KIND, EITHER EXPRESSED\n OR IMPLIED, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF\n MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE.  THE ENTIRE RISK AS\n TO THE QUALITY AND PERFORMANCE OF
THE PROGRAM IS WITH YOU.  SHOULD THE\n PROGRAM PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING,\n REPAIR OR CORRECTION.\n \n 12. IN NO
EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING\n WILL ANY
COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR\n REDISTRIBUTE
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,\n INCLUDING
ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING\n OUT OF
THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED\n TO LOSS
OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY\n YOU OR
THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER\n
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE\n
POSSIBILITY OF SUCH DAMAGES.\n \n END OF TERMS AND CONDITIONS");
	}

	/**
	 * Returns a short information about the License
	 *
	 * @return A String containing a short information about the used
license
	 *@since 0.0.2pA4
	 *@version 0
	 */
	protected String getLicenseInfo() {
		return ("This Software comes without any warranty.\n"
				+ "You may redistribute and/or modify it under the
terms of the GPL v 2.\n"
				+ "For details type 'license'.\n");
	}

	/**
	 * Returns the copyright informations valid for this software
	 *
	 * @return A String containing copyright informations
	 *@since 0.0.2pA4
	 *@version 0
	 */
	protected String getCopyright() {
		return ("Copyright 2000 - 2001 The JavaFTP-Group.\n"
				+ "For details type 'info'.\n");
	}

	/**
	 * Easter egg
	 *
	 * @since 0.0.2pA4
	 *@version [.,xX@Xx,.]
	 */
	protected void printLongJFTPInfo() {
		final boolean z = false;
		int i = ((!!!!!z) ? (-1) : (0));
		try {
			for (;;) {
				Thread.sleep(((!!!!!z) ? (0) : (1)), ((!!!!z) ?
(0) : (1)));
				(new
```

```
StdOut()).print(getLongJFTPInfo().charAt(++i));
                            if (i >= getLongJFTPInfo().length())
                                  break;
                      }
              } catch (Exception e) {
              }
       }
}

/*
 * Name:    JFTPSuperInterface.java  [INTERFACE]
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 17.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Nothing.
 *
 * @since 0.0.2pA1
 */
public interface JFTPSuperInterface {
}

/*
 * Name:    Listable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 06.05.2001 Tobias Kranz Rectifing comments
 * 0.1.99t2 17.07.2001 Tobias Kranz Added 4 methods.
 * 0.1.99t4 19.07.2001 Tobias Kranz Added 2 ~'printWorkingDir' methods.
 */
package jftp;

/**
 * Defines methods to list directories.
 *
 * @since 0.0.2pA1
 */
public interface Listable extends JFTPSuperInterface {
       /**
        * Prints the current working directory on the server.
        *
        * @return A java.lang.String containing the directory
        *@since 0.1.99t4
        *@version 1
        */
       public String printRemoteWorkingDir();

       /**
        * Prints the current working directory on the local host.
        *
        * @return A java.lang.String containing the directory
        *@since 0.1.99t4
        *@version 1
        */
       public String printLocalWorkingDir();
```

```
	/**
	 * Returns a detailed directory listing of the active directory.
	 *
	 * @return Directorylist of the active directory
	 *@since 0.0.2pA1
	 */
	public String getLongDirList();

	/**
	 * Returns a short directory listing of the active directory.
	 *
	 * @return Directorylist of the active directory
	 *@since 0.0.2pA1
	 */
	public String getShortDirList();

	/**
	 * Returns a short directory listing of the active directory in a
	 * jav.lang.String array.
	 *
	 * @return A java.lang.String array containing the Directorylist of
the
	 *         active directory
	 *@since 0.1.99t2
	 *@version 1
	 */
	public String[] getDirListArray();

	/**
	 * Returns a detailed directory listing of the active working-
directory.
	 *
	 * @return Directorylist of the active working-directory
	 *@since 0.0.2pA1
	 */
	public String getLocalLongDirList();

	/**
	 * Returns a short directory listing of the active working-directory.
	 *
	 * @return Directorylist of the active working-directory
	 *@since 0.0.2pA1
	 */
	public String getLocalShortDirList();

	/**
	 * Returns a short directory listing of the active working-directory
in a
	 * jav.lang.String array.
	 *
	 * @return A java.lang.String array containing the Directorylist of
the
	 *         active working-directory
	 *@since 0.1.99t2
	 *@version 1
	 */
	public String[] getLocalDirListArray();
}

/*
 * Name:    NetIO.java (extends JFTPIO)
 * Author:  JavaFTP-Group
```

```
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Class for network-io.
 *
 * @since 0.0.2pA1
 */
public class NetIO extends JFTPIO {
}

/*
 *  Name:        NetReader.java  (extends  NetIO  (extends  JFTPIO  (extends
JFTPSuper)))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.1    12.03.2001 Tobias Kranz creation.
 *          13.03.2001 Tobias Kranz changed from Thread to regular class
 *                                  (nearly total rewrite).
 * 0.0.1pA3 15.03.2001 Tobias Kranz splitted from JFTP; nothing big.
 * 0.0.1pA6 03.04.2001 Tobias Kranz added comments.
 * 0.0.1pA7 04.04.2001 Tobias Kranz splitted 'read()' to readLine() and
 *                                  readBlock().
 * 0.0.2pA2 03.05.2001 Tobias Kranz (re)added 'read()' which implements now
 *                                  'readLine()' & 'readBlock()'
 *   0.0.2pA4   15.05.2001   Tobias   Kranz   removed   everything!   created
'readStream()'
 *                                   which must be wrapped by 'receiveLine()'
&&
 *                                  'receiveBlock()' in FTPCmdServer.java .
 *                                  See README.DEVELOPMENT for details.
 * 0.1.99   10.06.2001 Tobias Kranz added 'getInputStream()'
 */

package jftp;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Socket;

/**
 * Class NetReader may be used to read any kind of data through an existing
 * Socket.
 *
 * @since 0.0.1
 */
public class NetReader extends NetIO {
       private InputStream inStream;
       private StdOut stdout = new StdOut();
       private StdErr stderr = new StdErr();

       /**
        * Sets the InputStream.
        *
        * @param socket
        *               Socket to listen on (This does NOT mean a listening
socket!).
```

```
         *@since 0.0.1
         */
        public NetReader(Socket socket) {
                try {
                        this.inStream = socket.getInputStream();
                } catch (Exception e) {
                        stderr.println("Unable    to    get    InputStream    from
Socket.");
                        stdout.println("Unable    to    get    InputStream    from
Socket.");
                }
        }

        /*
         * Get the stuff in.
         */
        /**
         * Autodetects the number of lines to read and give them back.
         *
         * @return String read from server
         *@since 0.0.2pA2
         */
        /*
         * public String read() { String rc = "", line; int i;
         *
         * InputStreamReader isr = new InputStreamReader(inStream);
         *
         * try { do // start reading the whole block (or just a line..) { line
= "";
         *
         * do // beginn reading the line { i = isr.read();/? should we make
         * 'isr.ready()' call(s) ? line += (char)i; } while (i != 10 ); //
until
         * '(char)10' (means CR (0x0A)) is read
         *
         * rc += line; // adding the read line to the return value
         *
         * } while ((char)line.charAt(3) != ' '); } catch (IOException ioe)
{ /
         * shall we do something here or should we throw this Exception ?? }
         *
         * return( rc.substring(0, (rc.length() - 1)) ); // trim last char
('\n') }
         */
        /**
         * Returns a InputStreamReader reading from the InputStream of the
given
         * Socket.
         *
         * @return InputStreamReader
         *@since 0.0.2pA4
         */
        public InputStreamReader getStream() {
                return (new InputStreamReader(inStream));
        }

        /**
         * Returns the InputStream of the used socket
         *
         * @return An InputStream
         *@since 0.1.99
         *@version 0
         */
```

```
	public InputStream getInputStream() {
		return (inStream);
	}

	/**
	 * Reads a single line of data from the server .
	 *
	 * @return Whatever the ftpserver sends us.
	 *@since 0.0.1
	 *@deprecated
	 */
	/*
	 * public String readLine() { return( this.read() ); } // just to stay
	 * compatible
	 */

	/**
	 * Reads a whole block of data from the server.
	 *
	 * @return Whatever the server sends.
	 *@since 0.0.1
	 *@deprecated
	 */
	/*
	 * public String readBlock() { String rc = ""; boolean done = false;
int i;
	 * //char read from server
	 *
	 * try { InputStreamReader isr = new InputStreamReader(inStream);
	 *
	 * while (! done) { i = isr.read();
	 *
	 * if (i == -1) // -1 ~ EOF { if (jftpSuper.getDebugLevel() >= 2)
	 * stdout.println("finished reading block"); done = true; } else { rc
+=
	 *	(char)i;	if	(jftpSuper.getDebugLevel()	>=	2)
stdout.println("Reading " +
	 * rc); } } } catch (Exception e) {
	 *	 stderr.println("Exception	while	reading	from
server."+e.toString()); }
	 *
	 * return rc; }
	 */
}

/*
 *  Name:        NetWriter.java  (extends  NetIO  (extends  JFTPIO  (extends
JFTPSuper)))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.1    12.03.2001 Tobias Kranz Creation
 *          13.03.2001 Tobias Kranz Changed from Thread to regular class
 *                                  (nearly total rewrite)
 * 0.0.1pA3 15.03.2001 Tobias Kranz splitted from JFTP; nothing big
 * 0.1.99   10.06.2001 Tobias Kranz added 'getOutputStream()'
 */

package jftp;

import java.io.OutputStream;
import java.net.Socket;
```

```
/**
 * Class NetWriter may be used to send any kind of data through an existing
 * Socket.
 *
 * @since 0.0.1
 */
public class NetWriter extends NetIO {
	private OutputStream outStream;
	private StdOut stdout = new StdOut();
	private StdErr stderr = new StdErr();

	/**
	 * Sets the OutputStream.
	 *
	 * @param socket
	 *            Socket to write through.
	 *@since 0.0.1
	 */
	public NetWriter(Socket socket) {
		try {
			this.outStream = socket.getOutputStream();
		} catch (Exception e) {
		}
	}

	/**
	 * Returns the OutputStream of the used socket
	 *
	 * @return The OutputStream used for the current socket.
	 *@since 0.1.99
	 *@version 0
	 */
	public OutputStream getOutputStream() {
		return (outStream);
	}

	/*
	 * Just send everything out to the server
	 */
	/**
	 * Writes data to the ftpserver
	 *
	 * @param string
	 *            The data to send
	 *@since 0.0.1
	 */
	public void write(String string) {
		int i;

		try {
			if (jftpSuper.getDebugLevel() >= 1)
				stdout.println("starting new write-loop");

			for (i = 0; i < string.length(); i++) {
				outStream.write(string.charAt(i));
				if (jftpSuper.getDebugLevel() >= 2)
					stdout.println("writing    char:    "    +
string.charAt(i));
			}
			if (string.charAt(string.length() - 1) != '\n')
				outStream.write('\n'); // Finish the line :-)
			else if (jftpSuper.getDebugLevel() >= 2)
```

```
                        stderr.println("DON'T SEND NEWLINES!");

                    outStream.flush(); // make sure the chars are written

                    if (jftpSuper.getDebugLevel() >= 1)
                        stdout.println("ending write-loop");
                } catch (Exception e) {
                    stderr.println("Exception in Outp(" + e.toString() +
")");
                }
        }
}

/*
 * Name:    Parser.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Abstract class. Defines methods to parse Strings.
 *
 * @since 0.0.2pA1
 */
public abstract class Parser extends JFTPSuper {
        public Parser() {
                boolean busy = false;
                System.out.println(busy);
                /*
                 * There was an error msg at compile-time that let me do such
strange
                 * things. ;-)
                 */
        }

        abstract void parse(String s);
}

/*
 * Name:    Progressbar.java  (extens JFTPSuper.java)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99   11.06.2001 Tobias Kranz Creation
 * 0.1.99t5 23.07.2001 Tobias Kranz Added Comments
 */
package jftp;

/**
 * Shows a textual progressbar
 *
 * @since 0.1.99
 *@version 1
 */
public class Progressbar extends JFTPSuper {
        private long max;
        private long pos = 1;
```

```
	/**
	 * Constructor. Here you must set the maximum value the progressbar
may
	 * have.
	 *
	 * @param max
	 *            The maximum value the progressbar may have.
	 *@since 0.1.99
	 *@version 1
	 */
	public Progressbar(int max) {
		this.max = max;
		this.show();
	}

	/**
	 * Sets the progressbar to the given value.
	 *
	 * @param pos
	 *            The current position of the progressbar
	 *@since 0.1.99
	 *@version 1
	 */
	protected void set(int pos) {
		if (pos <= max) {
			this.pos = pos;
		}
		show();
	}

	/**
	 * Shows the progressbar in it's current status
	 *
	 * @since 0.1.99
	 *@version 1
	 */
	private void show() {
		int i, stat = 0;
		String graphStat = "";
		double stat1 = ((double) 100 / max) * pos;

		String statstr = Double.toString(stat1);
		stat           =           Integer.parseInt(statstr.substring(0,
statstr.indexOf('.')));

		for (i = 0; i < (int) 20 * (((float) stat) / 100); i++)
			graphStat += "="; // setting '='s for reached %

		for (; i < 20; i++)
			graphStat += " "; // filling the rest with ' 's

		if ((int) ((float) stat) / 100 >= 1)
			System.out.print("Progress: [" + graphStat + "]: " +
stat
					+ "% complete (" + (int) pos + " bytes
transfered)\n");
		// else if ((int)((float)stat)/10 >= 1)
		//  System.out.print("Progress: ["+graphStat+"]:    "  +stat+"%
complete ("+
		// (int)pos/1024+" kb)\r");
		else
			System.out.print("Progress: [" + graphStat + "]:    " +
stat
```

```
                                    + "% complete (" + (int) pos / 1024 + "
kb)\r");
       }
}

/*
 * Name:    Removeable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 19.07.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Defines methods to remove directories and files.
 *
 * @since 0.1.99t4
 */
public interface Removeable extends JFTPSuperInterface {
       /**
        * Removes a Directory on the server with the given name
        *
        * @return True if successfully removed; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean removeRemoteDir(String dirName);

       /**
        * Removes a Directory on the local fs with the given name
        *
        * @return True if successfully removed; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean removeLocalDir(String dirName);
}

/*
 * Name:    ServerResponseParser.java (extends Parser (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Parses the FTPServers responses.
 *
 * @since 0.0.2pA1
 */
public class ServerResponseParser extends Parser {
       /**
        * Parses a given String for known FTPServer-replies.
        *
        * @param s
        *            String to parse
        *@since 0.0.2pA1
        */
```

```
	public void parse(String s) {
		/*
		 *  if  (s.equals(xxxxxx))  {  Insert  known  FTPServer  replies
here. }
		 */
	}
}

/*
 * Name:    StdErr.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Represents the standard error channel (StdErr).<BR>
 * The trailing '*' will be removed in stable versions.
 *
 * @since 0.0.2pA1
 */
public class StdErr extends StdIO {
	/**
	 * Prints an object to StdErr.
	 *
	 * @param o
	 *            Object to print
	 *@since 0.0.2pA1
	 */
	public void print(Object o) {
		System.err.print("*" + o);
	}

	/**
	 * Prints a double to StdErr.
	 *
	 * @param d
	 *            double to print
	 *@since 0.0.2pA1
	 */
	public void print(double d) {
		System.err.print("*" + d);
	}

	/**
	 * Prints a float to StdErr.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void print(float f) {
		System.err.print("*" + f);
	}

	/**
	 * Prints a long to StdErr.
	 *
	 * @param l
	 *            long to print
```

```
         *@since 0.0.2pA1
         */
        public void print(long l) {
               System.err.print("*" + l);
        }

        /**
         * Prints an int to StdErr.
         *
         * @param i
         *            int to print
         *@since 0.0.2pA1
         */
        public void print(int i) {
               System.err.print("*" + i);
        }

        /**
         * Prints a short to StdErr.
         *
         * @param s
         *            short to print
         *@since 0.0.2pA1
         */
        public void print(short s) {
               System.err.print("*" + s);
        }

        /**
         * Prints a char to StdErr.
         *
         * @param c
         *            char to print
         *@since 0.0.2pA1
         */
        public void print(char c) {
               System.err.print("*" + c);
        }

        /**
         * Prints a byte to StdErr.
         *
         * @param b
         *            byte to print
         *@since 0.0.2pA1
         */
        public void print(byte b) {
               System.err.print("*" + b);
        }

        /**
         * Prints a boolean to StdErr.
         *
         * @param b
         *            boolean to print
         *@since 0.0.2pA1
         */
        public void print(boolean b) {
               System.err.print("*" + b);
        }

        /**
         * Prints an object and "\n\r" to StdErr.
```

```
	 *
	 * @param o
	 *            Object to print
	 *@since 0.0.2pA1
	 */
	public void println(Object o) {
		System.err.println("*" + o);
	}

	/**
	 * Prints a double and "\n\r" to StdErr.
	 *
	 * @param d
	 *            double to print
	 *@since 0.0.2pA1
	 */
	public void println(double d) {
		System.err.println("*" + d);
	}

	/**
	 * Prints a float and "\n\r" to StdErr.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void println(float f) {
		System.err.println("*" + f);
	}

	/**
	 * Prints a long and "\n\r" to StdErr.
	 *
	 * @param l
	 *            long to print
	 *@since 0.0.2pA1
	 */
	public void println(long l) {
		System.err.println("*" + l);
	}

	/**
	 * Prints a int and "\n\r" to StdErr.
	 *
	 * @param i
	 *            int to print
	 *@since 0.0.2pA1
	 */
	public void println(int i) {
		System.err.println("*" + i);
	}

	/**
	 * Prints a short and "\n\r" to StdErr.
	 *
	 * @param s
	 *            short to print
	 *@since 0.0.2pA1
	 */
	public void println(short s) {
		System.err.println("*" + s);
	}
```

```
	/**
	 * Prints a char and "\n\r" to StdErr.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void println(char c) {
		System.err.println("*" + c);
	}

	/**
	 * Prints a byte and "\n\r" to StdErr.
	 *
	 * @param b
	 *            byte to print
	 *@since 0.0.2pA1
	 */
	public void println(byte b) {
		System.err.println("*" + b);
	}

	/**
	 * Prints a boolean and "\n\r" to StdErr.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void println(boolean b) {
		System.err.println("*" + b);
	}
}

/*
 * Name:    StdIn.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.1.99t5 01.08.2001 Tobias Kranz added 'readPassword()'
 */
package jftp;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * Represents the standard input channel (StdIn).
 *
 * @since 0.0.2pA1
 */
public class StdIn extends StdIO {
	private BufferedReader br;

	public StdIn() {
		this.br = new BufferedReader(new InputStreamReader(System.in));
	}
```

```
	/**
	 * Reads a whole line of data from the standard input channel.
	 *
	 * @return The String beeing read.
	 *@throws IOException
	 *             if reading fails.
	 *@since 0.0.2pA1
	 */
	protected String readLine() throws IOException {
		return ((String) br.readLine());
	}

	/**
	 * Reads a single char of data from the standard input channel.
	 *
	 * @return The char beeing read.
	 *@throws IOException
	 *             if reading fails.
	 *@since 0.0.2pA1
	 */
	protected char readChar() throws IOException {
		return ((char) br.read());
	}

	/**
	 * Reads an int from the standard input channel.
	 *
	 * @return The int being read
	 *@throws IOException
	 *             if reading fails
	 *@since 0.0.2pA1
	 */
	protected int readInt() throws IOException {
		return (Integer.valueOf(br.readLine()).intValue());
	}

	/**
	 * Reads an InetAddress from the standard input channel.
	 *
	 * @return The InetAddress read
	 *@throws IOException
	 *             if reading fails
	 *@throws UnknownHostException
	 *             if host is unknown
	 *@since 0.0.2pA1
	 */
	protected InetAddress readInetAddress() throws IOException,
			UnknownHostException {
		return (InetAddress.getByName(br.readLine()));
	}

	/**
	 * Reads a password and dissables the terminal-echo.
	 *
	 * @return The Password read from StdIn
	 *@throws IOException
	 *             if reading fails
	 *@since 0.1.99t5
	 *@version 1
	 */
	protected String readPasswd() throws IOException {
		String rc = "";
```

```
              StringBuffer sb = new StringBuffer();
              // char c;//////////////////////////////////////

              /*
               * Font-Colors:
               *
               *  Font-Colors are from 29 - 39. Background-Colors beginning
with 40.
               *
               *  #  color  29  Gray  30  Black  31  DarkRed  32  DarkGreen  33
Brown/DarkOrange
               * 34 DarkBlue 35 Purple 36 Marine 37 Gray 38 White 39 Gray
               */

              // Set Font-Color to Black
              sb.append('\u001b');
              sb.append('[');
              sb.append(30); // Color
              sb.append('m');
              System.out.print(sb.toString()); // Set it.

              // read
              rc = br.readLine();

              // (re)Set Font-Color to Gray
              sb = new StringBuffer(); // Reset it
              sb.append('\u001b');
              sb.append('[');
              sb.append(39); // Color
              sb.append('m');
              System.out.print(sb.toString()); // Set it.

              return (rc);
       }
}

/*
 * Name:    StdIO.java (extends JFTPIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
package jftp;

/**
 * Class for standard-io
 *
 * @since 0.0.2pA1
 */
public class StdIO extends JFTPIO {
}

/*
 * Name:    StdOut.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 15.05.2001 Tobias Kranz fixed "double newline"-bug
 */
```

```
package jftp;

/**
 * Represents the standard output channel (StdOut)
 *
 * @since 0.0.2pA1
 */
public class StdOut extends StdIO {
	/**
	 * Prints an object to StdOut.
	 *
	 * @param o
	 *            Object to print
	 *@since 0.0.2pA1
	 */
	public void print(Object o) {
		if (jftpSuper.isCli()) {
			System.out.print(o);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a double to StdOut.
	 *
	 * @param d
	 *            double to print
	 *@since 0.0.2pA1
	 */
	public void print(double d) {
		if (jftpSuper.isCli()) {
			System.out.print(d);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a float to StdOut.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void print(float f) {
		if (jftpSuper.isCli()) {
			System.out.print(f);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a long to StdOut.
	 *
	 * @param l
	 *            long to print
	 *@since 0.0.2pA1
	 */
	public void print(long l) {
		if (jftpSuper.isCli()) {
			System.out.print(l);
```

```
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints an int to StdOut.
	 *
	 * @param i
	 *             int to print
	 *@since 0.0.2pA1
	 */
	public void print(int i) {
		if (jftpSuper.isCli()) {
			System.out.print(i);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a short to StdOut.
	 *
	 * @param s
	 *             short to print
	 *@since 0.0.2pA1
	 */
	public void print(short s) {
		if (jftpSuper.isCli()) {
			System.out.print(s);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a byte to StdOut.
	 *
	 * @param b
	 *             byte to print
	 *@since 0.0.2pA1
	 */
	public void print(byte b) {
		if (jftpSuper.isCli()) {
			System.out.print(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a char to StdOut.
	 *
	 * @param c
	 *             char to print
	 *@since 0.0.2pA1
	 */
	public void print(char c) {
		if (jftpSuper.isCli()) {
			System.out.print(c);
		} else {
			// Insert GUI-Code here
		}
```

```
	}

	/**
	 * Prints a boolean to StdOut.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void print(boolean b) {
		if (jftpSuper.isCli()) {
			System.out.print(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints an object and "\n\r" to StdOut.
	 *
	 * @param o
	 *            Object to print
	 *@since 0.0.2pA1
	 *@version 0
	 */
	public void println(Object o) {
		if (jftpSuper.isCli()) {
			if  ((char)  o.toString().charAt(o.toString().length()  -
1) == '\n')
				System.out.print(o);
			else
				System.out.println(o);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a double and "\n\r" to StdOut.
	 *
	 * @param d
	 *            double to print
	 *@since 0.0.2pA1
	 */
	public void println(double d) {
		if (jftpSuper.isCli()) {
			System.out.println(d);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a float and "\n\r" to StdOut.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void println(float f) {
		if (jftpSuper.isCli()) {
			System.out.println(f);
		} else {
```

```
                // Insert GUI-Code here
        }
    }

    /**
     * Prints a long and "\n\r" to StdOut.
     *
     * @param l
     *            long to print
     *@since 0.0.2pA1
     */
    public void println(long l) {
        if (jftpSuper.isCli()) {
            System.out.println(l);
        } else {
            // Insert GUI-Code here
        }
    }

    /**
     * Prints an int and "\n\r" to StdOut.
     *
     * @param i
     *            int to print
     *@since 0.0.2pA1
     */
    public void println(int i) {
        if (jftpSuper.isCli()) {
            System.out.println(i);
        } else {
            // Insert GUI-Code here
        }
    }

    /**
     * Prints a short and "\n\r" to StdOut.
     *
     * @param s
     *            short to print
     *@since 0.0.2pA1
     */
    public void println(short s) {
        if (jftpSuper.isCli()) {
            System.out.println(s);
        } else {
            // Insert GUI-Code here
        }
    }

    /**
     * Prints a byte and "\n\r" to StdOut.
     *
     * @param b
     *            byte to print
     *@since 0.0.2pA1
     */
    public void println(byte b) {
        if (jftpSuper.isCli()) {
            System.out.println(b);
        } else {
            // Insert GUI-Code here
        }
    }
```

```
	/**
	 * Prints a char and "\n\r" to StdOut.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void println(char c) {
		if (jftpSuper.isCli()) {
			System.out.println(c);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a boolean and "\n\r" to StdOut.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void println(boolean b) {
		if (jftpSuper.isCli()) {
			System.out.println(b);
		} else {
			// Insert GUI-Code here
		}
	}
}

/*
 * Name:    Transferable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 *  0.0.2pA2  03.05.2001  Tobias  Kranz  Changed  'receive-Line/-Block'  to
'receive()'
 * 0.1.0    09.06.2001 Martin Loh   Changed the 'getFile()' and 'putFile()'
 *                                          method definitions so that they
 *                                                return an boolean and have
Strings
 *                                          as args.
 * 0.1.99t2 17.07.2001 Tobias Kranz Added 'getFiles(String[], String[])',
 *                                  corrected some comments.
 * 0.1.99t4 19.07.2001 Tobias Kranz cleaned up
 */
package jftp;

/**
 * Defines methods to transfer "raw" data and files.
 *
 * @since 0.0.2pA1
 */
public interface Transferable extends JFTPSuperInterface {
	/**
	 * Reads data from the ctrl-channel.
	 *
	 * @return The data read.
	 *@since 0.0.2pA4
```

```
         *@version 0
         */
        public String readCtrl();

        /**
         * Receives a whole block of data.
         *
         * @return The block received
         *@since 0.0.2pA4
         *@version 0
         */
        public String readCtrlBlock();

        /**
         * Reads data until end of Stream
         *
         * @return The data read.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readData();

        /**
         * Reads a whole block of data.
         *
         * @return The block received
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataBlock();

        /**
         * Reads a single line of data.
         *
         * @return The data received.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataLine();

        /**
         * Recieves a list of Files 'serverlist' and saves it to 'localList'.
         *
         * @param serverList
         *            Filelist (java.lang.String array)
         *@param localList
         *            Filelist (java.lang.String array)
         *@return true if successfully transfered; otherwise false.
         *@since 0.1.99t2
         *@version 1
         */
        public boolean getFiles(String[] serverList, String[] localList);

        /**
         * Recieves a File with name 'Serverfile' and saves it to 'localfile'.
         *
         * @param ServerFile
         *            Filename
         *@param LocalFile
         *            Filename
         *@return true if successfully transfered; otherwise false.
         *@since 0.0.2pA1
         *@version 1
```

```
         */
        public boolean getFile(String ServerFile, String LocalFile);

        /**
         * Puts a File named 'LocalFile'.
         *
         * @param LocalFile
         *             Filename
         *@return true if the file was transfered successfully; false if not.
         *@since 0.0.2pA1
         *@version 0
         */
        public boolean putFile(String LocalFile);

        /**
         * Sends a string of data.
         *
         * @param s
         *             String to send.
         *@since 0.0.2pA1
         *@version 0
         */
        public void send(String s);
}
```

Source Code B-5: JFTP Original Source Code

```
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.net.InetAddress;

/**
 * 2.3.1. Replace Conditional with Polymorphism: FTPASCIIData extends
 * FTPDataType.
 */
public class FTPASCIIData extends FTPDataType {

	/**
	 * 2.3.2. FTPASCIIData constructors invoke FTPDataType constructors.
	 */
	public FTPASCIIData() {
		super();
	}

	public FTPASCIIData(InetAddress server, int port) {
		super(server, port);
	}

	/**
	 * 2.3.3. setTxMode method is overridden.
	 */
	public boolean setTxMode(String mode) {
		boolean modeIsValid = false;

		if (isASCIIMode(mode)) {
			modeIsValid = true;
			sendTxMode(mode);
		}

		if (modeIsValid)
			printOutputLine(readCtrl());

		return (modeIsValid);
	}

	/**
	 * 2.3.3. setTxMode method is overridden.
	 */
	public boolean setTxMode(char nm) {
		boolean rc = false;

		if (isASCIIMode(nm))
			rc = setTxMode(ASCII);

		return (rc);
	}

	/**
	 * 2.3.3. sendTxMode method is overridden.
	 */
	protected void sendTxMode(String mode) {
```

```
                if (isASCIIMode(mode)) {
                        send("type a");
                }
        }

        /**
         * 3.4. Decompose Conditional: isASCIIMode method with parameter of
String
         * type.
         */
        protected boolean isASCIIMode(String mode) {
                return ((mode.equals(ASCII)) || (mode.equals(ASC)));
        }

        /**
         * 3.4. Decompose Conditional: isASCIIMode method with parameter of
         * character type.
         */
        protected boolean isASCIIMode(char mode) {
                return ((mode == 'a'));
        }
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.io.IOException;
import java.net.InetAddress;

import
jftprefactoring.flexibilityandextensibility.interfaces.Authenticateable;

/**
 *  1.2.1.  Extract  Subclass:  FTPAuthentication  extends  FTPCmdServer  and
relevant
 * methods are pushed down. 1.4.2. FTPAuthentication extends FTPConnection.
 */
public class FTPAuthentication extends FTPConnection implements
                Authenticateable {

        /**
         * 1.3.1. Push Down Field: userName, anonymousPasswd and authenticated
are
         * pushed down with encapsulation methods from JFTPSuper.
         */
        private String userName = "me";
        private String anonymousPasswd = "tux@northpole.org";
        private boolean authenticated = false;

        /**
         * 1.3.3. FTPAuthentication constructors. 1.4.3. FTPAuthentication
         * constructors invoke FTPConnection constructor.
         */
        public FTPAuthentication() {
                super();
        }
```

```
	public FTPAuthentication(InetAddress server, int port) {
		super(server, port);
	}

	public FTPAuthentication(String userName, String anonymousPasswd,
			boolean authenticated) {
		super();
		this.userName = userName;
		this.anonymousPasswd = anonymousPasswd;
		this.authenticated = authenticated;
	}

	public FTPAuthentication(InetAddress server, int port, String
userName,
			String anonymousPasswd, boolean authenticated) {
		super(server, port);
		this.userName = userName;
		this.anonymousPasswd = anonymousPasswd;
		this.authenticated = authenticated;
	}

	/**
	 * 1.3.2. Encapsulation methods are pushed down with fields from
JFTPSuper.
	 */

	/**
	 * Returns the userName
	 *
	 * @return The current userName
	 *@since 0.0.2pA2
	 */
	public String getUserName() {
		return (userName);
	}

	/**
	 * Sets the userName
	 *
	 * @param s
	 *            New userName
	 *@since 0.0.2pA2
	 */
	public void setUserName(String s) {
		userName = s;
	}

	/**
	 * Returns the anonymous passwd.
	 *
	 * @return The password used for anonymous login.
	 *@since 0.0.2pA2
	 */
	public String getAnonymousPasswd() {
		return (anonymousPasswd);
	}

	/**
	 * Sets the password used for anonymous login to s.
	 *
	 * @param s
	 *            The password used for anonymous login.
	 *@since 0.0.2pA2
```

```
	 */
	public void setAnonymousPasswdTo(String s) {
		anonymousPasswd = s;
	}

	/**
	 * Are we authenticated ?
	 *
	 * @return true if we are authenticated; else false
	 *@since 0.0.2pA2
	 */
	public boolean isAuthenticated() {
		return (authenticated);
	}

	/**
	 * Sets authenticated to true
	 *
	 * @since 0.0.2pA2
	 */
	public void setAuthenticated() {
		authenticated = true;
	}

	/**
	 * Sets authenticated to false
	 *
	 * @since 0.0.2pA2
	 */
	public void unsetAuthenticated() {
		authenticated = false;
	}

	/**
	 * 1.2.2. Relevant methods are pushed down from FTPCmdServer.
	 */

	/* <START Implementing "Authenticateable"> */

	/**
	 * Authenticates an user
	 *
	 * @return true if user is authenticated; otherwise false
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public boolean authenticate() {
		boolean rc = true;
		String passwd = "", strReceive;

		try {
			unsetAuthenticated();
			printOutput("Username: ");
			setUserName(readInputLine()); // shouldn't we check
			// this?

			send("user " + getUserName());
			strReceive = readCtrl();
			printOutputLine(strReceive);
			if (usernameIsOkay(strReceive)) {
				if (getUserName().equals("anonymous")) {
					passwd = getAnonymousPasswd();
				} else {
```

```
                                printOutput("Password: ");
                        }

                        try {
                                passwd = readInputPassword();
                        } catch (IOException ioe) {
                                printErrorLine("Unable    to    read    the
password.");
                        }

                        send("pass " + passwd);

                        if (getDebugLevel() >= 2)
                                printErrorLine("User: " + getUserName() + "
Pwd: " + passwd);

                        strReceive = this.readCtrl();
                        if (!userIsLoggedIn(strReceive)) {
                                rc = false;
                        }
                        printOutputLine(strReceive);
                } else {
                        printErrorLine("Login failed.");
                        rc = false;
                }
        } catch (Exception e) {
                rc = false;
                //  getStderr.println("Unable  to  authenticate.");//Only
debug(tk)
        }

        return (rc);
    }

    /* <END Implementing "Authenticateable"> */

    /**
     * 3.4. Decompose Conditional: usernameIsOkay method.
     */
    private boolean usernameIsOkay(String response) {
            return (response.startsWith("331"));
    }

    /**
     * 3.4. Decompose Conditional: userIsLoggedIn method.
     */
    private boolean userIsLoggedIn(String response) {
            return (response.startsWith("230"));
    }
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.net.InetAddress;

import
```

```
jftprefactoring.flexibilityandextensibility.interfaces.DataTypeChangeable;

/**
 * 1.2.1. Extract Subclass: FTPDataType extends FTPCmdServer and relevant
 * methods are pushed down. 1.4.2. FTPDataType extends FTPConnection. 2.3.4.
 * FTPDataType is converted to abstract class.
 */
public abstract class FTPDataType extends FTPConnection implements
            DataTypeChangeable {

      /**
       * 1.8.1. Replace Magic Number with Symbolic Constant: ASCII, ASC,
BINARY
       * and BIN.
       */
      protected String ASCII = "ascii";
      protected String ASC = "asc";
      protected String BINARY = "binary";
      protected String BIN = "bin";

      /**
       * 1.3.3. FTPDataType constructors. 1.4.3. FTPDataType constructors
invoke
       * FTPConnection constructor.
       */
      public FTPDataType() {
            super();
      }

      public FTPDataType(InetAddress server, int port) {
            super(server, port);
      }

      /**
       * 1.2.2. Relevant methods are pushed down from FTPCmdServer.
       */

      /* <START Implementing "Changeable"> */

      /**
       * 1.8.2. ASCII, ASC, BINARY and BIN replace their values setTxMode.
2.3.5.
       * setTxMode method is converted to abstract method.
       */

      /**
       * Sets the transfermode either to ascii or to binary
       *
       * @param mode
       *            The mode to use ("ascii" or "binary")
       *@return true if mode is set, otherwise false
       *@since 0.1.99
       *@version 0
       */
      public abstract boolean setTxMode(String mode);

      /**
       * 1.8.2. ASCII, and BINARY replace their values setTxMode. 2.3.5.
setTxMode
       * method is converted to abstract method.
       */
      public abstract boolean setTxMode(char nm);
```

```
	/* <END Implementing "Changeable"> */

	/**
	 * 1.6. Extract Method: sendTxMode method. 2.3.5. sendTxMode method is
	 * converted to abstract method.
	 */
	protected abstract void sendTxMode(String mode);
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.net.InetAddress;

/**
 * 2.3.1. Replace Conditional with Polymorphism: FTPBinaryData extends
 * FTPDataType.
 */
public class FTPBinaryData extends FTPDataType {

	/**
	 * 2.3.2. FTPBinaryData constructors invoke FTPDataType constructors.
	 */
	public FTPBinaryData() {
		super();
	}

	public FTPBinaryData(InetAddress server, int port) {
		super(server, port);
	}

	/**
	 * 2.3.3. setTxMode method is overridden.
	 */
	public boolean setTxMode(String mode) {
		boolean modeIsValid = false;

		if (isBinaryMode(mode)) {
			modeIsValid = true;
			sendTxMode(mode);
		}

		if (modeIsValid)
			printOutputLine(readCtrl());

		return (modeIsValid);
	}

	/**
	 * 2.3.3. setTxMode method is overridden.
	 */
	public boolean setTxMode(char nm) {
		boolean rc = false;

		if (isBinaryMode(nm))
			rc = setTxMode(BINARY);
```

```
                return (rc);
        }

        /**
         * 2.3.3. sendTxMode method is overridden.
         */
        protected void sendTxMode(String mode) {
                if (isBinaryMode(mode)) {
                        send("type i");
                }
        }

        /**
         * 3.4. Decompose Conditional: isBinaryMode method with parameter of
String
         * type.
         */
        protected boolean isBinaryMode(String mode) {
                return ((mode.equals(BINARY)) || (mode.equals(BIN)));
        }

        /**
         * 3.4. Decompose Conditional: isBinaryMode method with parameter of
         * character type.
         */
        protected boolean isBinaryMode(char mode) {
                return ((mode == 'i') || (mode == 'b'));
        }
}

/*
 * Name:    FTPCmdServer.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name               changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz       Creation
 * 0.0.2pA2 02.05.2001 Svenja Wittstadt   authenticate method implemented
 * 0.0.2pA3 06.05.2001 Tobias Kranz       added skeleton of 'int getPort()'.
 * 0.0.2pA4 15.05.2001 Tobias Kranz       changed 'String receiveLine()' &
 *                                        'String receiveBlock()'.
 * 0.0.2pA4 28.05.2001 Tobias Kranz       some small fixes...
 * 0.0.2pA4 06.06.2001 Tobias Kranz       REWRITE from scratch
 * 0.1.0    07.06.2001 Tobias Kranz &&
 *                     Sebastian Schipper Completed the rewrite.
 *                                        EVERYTHING´s FINE! ;-]
 * 0.1.0    09.06.2001 Martin Loh         Implemented the putFile Method and
 *                                        updated the getFile method
 * 0.1.99    10.06.2001 Tobias Kranz          Fixed bugs: cantListAfterPut,
cantGet.
 *                                        add 'setTxMode(String)' &
 *                                        'getFileSize(String)'
 * 0.1.99t1 11.06.2001 Tobias Kranz         added Progressbar in 'putFile()'
&&
 *                                        'getFile()'.
 * 0.1.99t1 12.06.2001 Tobias Kranz       added 'getFiles(String regex)'
 *                                            'not sure if this is the right
place..
 * 0.1.99t1 19.06.2001 Tobias Kranz       improved 'getFileSize(String)' I
 * 0.1.99t2 17.07.2001 Tobias Kranz       improved 'getFileSize(String)' II
 *                                        added 'getDirListArray()' &&
 *                                        'getLocalDirListArray()' &&
 *                                        'getLocalLongDirList()' &&
```

```
 *                                          'getLocalShortDirList()'
 * 0.1.99t4 19.07.2001 Tobias Kranz       Made an improvement suggestion in
 *                                                  'getFile()' && started to
implement
 *                                                    the "Createable" && the
"Removeable"
 *                                          interfaces.
 * 0.1.99t5 23.07.2001 Tobias Kranz        changed 'getFile()' && 'putFile()'
to
 *                                          work in the local working dir
 * 0.1.99t5 01.08.2001 Tobias Kranz       changed 'authenticate()'
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;

import jftprefactoring.flexibilityandextensibility.io.NetReader;
import jftprefactoring.flexibilityandextensibility.io.NetWriter;
import jftprefactoring.flexibilityandextensibility.io.StdErr;
import jftprefactoring.flexibilityandextensibility.io.StdIn;
import jftprefactoring.flexibilityandextensibility.io.StdOut;

/**
 * 1.2.0. Remove final keyword from FTPCmdServer.
 */

/**
 * This class contains the implementation of all supported FTP-Commands
 *
 * @since 0.0.2pA1
 */
public class FTPCmdServer extends JFTPSuper {

	/**
	 * Instance Variables
	 */
	private Socket ctrlSock;
	private Socket dataSock;

	private NetReader ctrlReader;
	private NetWriter ctrlWriter;
	private NetReader dataReader;
	private NetWriter dataWriter;

	private InputStream dataIs;
	private OutputStream dataOs;

	private InputStreamReader ctrlIsr;
	private InputStreamReader dataIsr;

	private StdIn stdin = new StdIn();
	private StdOut stdout = new StdOut();
	private StdErr stderr = new StdErr();
```

```
	/**
	 * 1.1. Encapsulate Fields: getCtrlSock and setCtrlSock.
	 */

	/**
	 * @return the ctrlSock
	 */
	public Socket getCtrlSock() {
		return ctrlSock;
	}

	/**
	 * @param ctrlSock
	 *            the ctrlSock to set
	 */
	public void setCtrlSock(Socket ctrlSock) {
		this.ctrlSock = ctrlSock;
	}

	/**
	 * 1.1. Encapsulate Fields: getDataSock and setDataSock.
	 */

	/**
	 * @return the dataSock
	 */
	public Socket getDataSock() {
		return dataSock;
	}

	/**
	 * @param dataSock
	 *            the dataSock to set
	 */
	public void setDataSock(Socket dataSock) {
		this.dataSock = dataSock;
	}

	/**
	 * 1.1. Encapsulate Fields: getCtrlReader and setCtrlReader.
	 */

	/**
	 * @return the ctrlReader
	 */
	public NetReader getCtrlReader() {
		return ctrlReader;
	}

	/**
	 * @param ctrlReader
	 *            the ctrlReader to set
	 */
	public void setCtrlReader(NetReader ctrlReader) {
		this.ctrlReader = ctrlReader;
	}

	/**
	 * 1.1. Encapsulate Fields: getCtrlWriter and setCtrlWriter.
	 */

	/**
```

```
	 * @return the ctrlWriter
	 */
	public NetWriter getCtrlWriter() {
		return ctrlWriter;
	}

	/**
	 * @param ctrlWriter
	 *            the ctrlWriter to set
	 */
	public void setCtrlWriter(NetWriter ctrlWriter) {
		this.ctrlWriter = ctrlWriter;
	}

	/**
	 * 1.1. Encapsulate Fields: getDataReader and setDataReader.
	 */

	/**
	 * @return the dataReader
	 */
	public NetReader getDataReader() {
		return dataReader;
	}

	/**
	 * @param dataReader
	 *            the dataReader to set
	 */
	public void setDataReader(NetReader dataReader) {
		this.dataReader = dataReader;
	}

	/**
	 * 1.1. Encapsulate Fields: getDataWriter and setDataWriter.
	 */

	/**
	 * @return the dataWriter
	 */
	public NetWriter getDataWriter() {
		return dataWriter;
	}

	/**
	 * @param dataWriter
	 *            the dataWriter to set
	 */
	public void setDataWriter(NetWriter dataWriter) {
		this.dataWriter = dataWriter;
	}

	/**
	 * 1.1. Encapsulate Fields: getDataIs and setDataIs.
	 */

	/**
	 * @return the dataIs
	 */
	public InputStream getDataIs() {
		return dataIs;
	}
```

```
	/**
	 * @param dataIs
	 *            the dataIs to set
	 */
	public void setDataIs(InputStream dataIs) {
		this.dataIs = dataIs;
	}

	/**
	 * 1.1. Encapsulate Fields: getDataOs and setDataOs.
	 */

	/**
	 * @return the dataOs
	 */
	public OutputStream getDataOs() {
		return dataOs;
	}

	/**
	 * @param dataOs
	 *            the dataOs to set
	 */
	public void setDataOs(OutputStream dataOs) {
		this.dataOs = dataOs;
	}

	/**
	 * 1.1. Encapsulate Fields: getCtrlIsr and setCtrlIsr.
	 */

	/**
	 * @return the ctrlIsr
	 */
	public InputStreamReader getCtrlIsr() {
		return ctrlIsr;
	}

	/**
	 * @param ctrlIsr
	 *            the ctrlIsr to set
	 */
	public void setCtrlIsr(InputStreamReader ctrlIsr) {
		this.ctrlIsr = ctrlIsr;
	}

	/**
	 * 1.1. Encapsulate Fields: getDataIsr and setDataIsr.
	 */

	/**
	 * @return the dataIsr
	 */
	public InputStreamReader getDataIsr() {
		return dataIsr;
	}

	/**
	 * @param dataIsr
	 *            the dataIsr to set
	 */
	public void setDataIsr(InputStreamReader dataIsr) {
		this.dataIsr = dataIsr;
```

```
	}

	/**
	 * 1.1. Encapsulate Fields: getStdin and setStdin. 1.7. Remove Setting
	 * Method: setStdin method is removed because it is set at creation
time and
	 * never altered.
	 */

	/**
	 * @return the stdin
	 */
	public StdIn getStdin() {
		return stdin;
	}

	/**
	 * @param stdin
	 *            the stdin to set
	 */
	/*
	 * public void setStdin(StdIn stdin) { this.stdin = stdin; }
	 */

	/**
	 *  1.1.  Encapsulate  Fields:  getStdout  and  setStdout.  1.7.  Remove
Setting
	 * Method: setStdout method is removed because it is set at creation
time
	 * and never altered.
	 */

	/**
	 * @return the stdout
	 */
	public StdOut getStdout() {
		return stdout;
	}

	/**
	 * @param stdout
	 *            the stdout to set
	 */
	/*
	 *
	 * public void setStdout(StdOut stdout) { this.stdout = stdout; }
	 *//**
	 *  1.1.  Encapsulate  Fields:  getStderr  and  setStderr.  1.7.  Remove
Setting
	 * Method: setStderr method is removed because it is set at creation
time
	 * and never altered.
	 */

	/**
	 * @return the stderr
	 */
	public StdErr getStderr() {
		return stderr;
	}

	/**
	 * @param stderr
```

```
         *            the stderr to set
         */
        /*
         * public void setStderr(StdErr stderr) { this.stderr = stderr; }
         */

        /**
         *  1.2.2.  All  common  methods  are  pushed  down  to  corresponding
extracted
         * subclass from FTPCmdServer.
         */
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.io.IOException;
import java.net.BindException;
import java.net.ConnectException;
import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.NoRouteToHostException;
import java.net.ProtocolException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.net.UnknownServiceException;
import java.util.BitSet;
import java.util.StringTokenizer;

import jftprefactoring.flexibilityandextensibility.interfaces.Connectable;
import jftprefactoring.flexibilityandextensibility.io.NetReader;
import jftprefactoring.flexibilityandextensibility.io.NetWriter;
import jftprefactoring.flexibilityandextensibility.io.StdErr;

/**
 * 1.2.1. Extract Subclass: FTPConnection extends FTPCmdServer and relevant
 * methods are pushed down. 1.4.1. Extract Super-Class: FTPConnection is set
to
 * be super class for other FTPCmdServer subclasses.
 */
public class FTPConnection extends FTPCmdServer implements Connectable {

        /**
         * 1.3.1. Push Down Field: server and port are pushed down with
         * encapsulation methods from JFTPSuper.
         */
        private InetAddress server;
        private int port = 21;

        /**
         * 1.3.1. Push Down Field: connected and passiveConnected options are
pushed
         * down with encapsulation methods from JFTPSuper.
         */
        private BitSet options = new BitSet(2);

        /**
```

```
        * 1.3.3. FTPConnection constructors.
        */
       public FTPConnection() {
              super();
              try {
                     this.server = InetAddress.getByName("localhost");
              } catch (UnknownHostException uhe) {
                     System.err.println("COUGHT!");
              }
              /*
               * BitSet 'options'
               *
               *  #:  if  true:  default:  0  =>  connected  false,  1  =>
passiveConnected
               * false.
               */
              this.options.clear(0);
              this.options.clear(1);
       }

       public FTPConnection(InetAddress server, int port) {
              super();
              this.server = server;
              this.port = port;
       }

       /**
        * 1.3.2. Encapsulation methods are pushed down with fields from
JFTPSuper.
        */

       /**
        * Returns the current Server
        *
        * @return InetAddress of the Server
        *@since 0.0.2pA1
        */
       public InetAddress getServer() {
              return (server);
       }

       /**
        * Returns the IP of the Server
        *
        * @return the servers IP
        *@since 0.0.2pA1
        */
       public String getServerIP() {
              return (server.getHostAddress());
       }

       /**
        * Returns the Name of the Server
        *
        * @return the servers Name
        *@since 0.0.2pA1
        */
       public String getServerName() {
              return (server.getHostName());
       }

       /**
        * sets the Servers IP/Name
```

```
         *
         * @param s
         *            Servers name/IP
         *@return true if the Server can be set; otherwise false
         *@version 2
         *@since 0.0.2pA1
         */
        public boolean setServerTo(String s) {
               boolean rc;

               try {
                      server = InetAddress.getByName(s);

                      rc = true;
               } catch (UnknownHostException uhe) {
                      StdErr stderr = new StdErr();
                      stderr.println("Unable to reach host '" + s + "'.");
                      rc = false;
               }

               return (rc);
        }

        /**
         * What's the current port?
         *
         * @return The current port
         *@since 0.0.2pA1
         */
        public int getServerPort() {
               return (port);
        }

        /**
         * Sets the port
         *
         * @param port
         *            Port
         *@since 0.0.2pA1
         */
        public void setServerPortTo(int port) {
               this.port = port;
        }

        /**
         *  1.3.2.  Encapsulation  methods  are  pushed  down  with  fields  from
JFTPSuper.
         */

        /**
         * Are we connected to a ftpserver ?
         *
         * @return true if connected; otherwise false
         *@since 0.0.2pA1
         */
        public boolean isConnected() {
               return ((boolean) options.get(0));
        }

        /**
         * Sets connected to true.
         *
         * @since 0.0.2pA1
```

```
         */
        protected void setConnected() {
                options.set(0);
        }

        /**
         * Sets connected to false.
         *
         * @since 0.0.2pA1
         */
        protected void unsetConnected() {
                options.clear(0);
        }

        /**
         * Are we passive Connected
         *
         * @return true if we are passive connected; otherwise false
         *@since 0.0.2pA4
         *@version 0
         */
        protected boolean isPassiveConnected() {
                return (options.get(1));
        }

        /**
         * Sets passive connected to true
         *
         * @since 0.0.2pA4
         *@version 1
         */
        protected void setPassiveConnected() {
                options.set(1);
        }

        /**
         * Sets passive connected to false
         *
         * @since 0.0.2pA4
         *@version 1
         */
        protected void unsetPassiveConnected() {
                options.clear(1);
        }

        /**
         * 1.5. Hide Delegate: FTPTransfer is hidden by readCtrl and send
methods.
         */
        protected String readCtrl() {
                FTPTransfer transfer = new FTPTransfer(this.getServer(), this
                                .getServerPort());
                return transfer.readCtrl();
        }

        protected void send(String s) {
                FTPTransfer transfer = new FTPTransfer(this.getServer(), this
                                .getServerPort());
                transfer.send(s);
        }

        /**
         * 1.2.2. Relevant methods are pushed down from FTPCmdServer.
```

```
         */

        /* <START Implementing "Connectable"> */

        /**
         * Opens a connection
         *
         * @return true if connected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean connect() {
                // return (openCtrlConnection());
                return (connect('c'));
        }

        /**
         * 3.2.1. Parameterize Method: connect method is overloaded by
         *  parameterizing  it  with  connection  type  flag  where  c  represents
control
         *  connection  and  d  represents  passive  data  connection.  3.2.2.
Parameterized
         * connect method is invoked in original connect method.
         */
        public boolean connect(char type) {
                if (type == 'c')
                        return (openCtrlConnection());
                else if (type == 'd')
                        return (openPassiveDataConnection());
                else
                        return false;
        }

        /**
         * Opens a ctrl connection
         *
         * @return true if connected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean openCtrlConnection() {
                boolean rc = false;

                if (prepareCtrlConnection()) {
                        if (getDebugLevel() >= 2)
                                printErrorLine("Trying  to  get  Streams  from  the
ctrlSock.");

                        setCtrlReader(new NetReader(getCtrlSock()));
                        setCtrlWriter(new NetWriter(getCtrlSock()));

                        setConnected();
                        rc = true;
                } else {
                        printErrorLine("Unable to connect.");
                }

                return (rc);
        }

        /**
         * 3.4. Decompose Conditional: prepareCtrlConnection method.
         */
```

```
	private boolean prepareCtrlConnection() {
		return (!(isConnected()) && (bindCtrlSocket()));
	}

	/**
	 * Closes a connection
	 *
	 * @return true if disconnected; false if not
	 *@since 0.1.0
	 *@version 0
	 */
	public boolean disconnect() {
		// return (closeCtrlConnection());
		return (disconnect('c'));
	}

	/**
	 * 3.2.1. Parameterize Method: disconnect method is overloaded by
	 *  parameterizing it with connection type flag where c represents
control
	 *  connection and d represents passive data connection. 3.2.2.
Parameterized
	 * disconnect method is invoked in original disconnect method.
	 */
	public boolean disconnect(char type) {
		if (type == 'c')
			return (closeCtrlConnection());
		else if (type == 'd')
			return (closePassiveDataConnection());
		else
			return false;
	}

	/**
	 * Closes a ctrl connection
	 *
	 * @return true if disconnected; false if not
	 *@since 0.1.0
	 *@version 0
	 */
	public boolean closeCtrlConnection() {
		boolean rc = false;

		if (isConnected()) {
			send("QuiT");
			printOutputLine(readCtrl());

			try {
				getCtrlSock().close();
				setCtrlSock(null);
				setCtrlIsr(null);
				setCtrlReader(null);
				setCtrlWriter(null);

				rc = true;
			} catch (IOException ioe) {
				writeLineToConsole("Exception    while    closing
ControlSocket.");
			}

			unsetConnected();
		}
```

```
		return (rc);
	}

	/**
	 * Opens a passive connection for data.
	 *
	 * @return true if connected; false if not
	 *@since 0.1.0
	 *@version 0
	 */
	public boolean openPassiveDataConnection() {
		int port;
		boolean rc = true;
		String receive;

		if (preparePassiveDataConnection()) {
			unsetPassiveConnected();

			send("pasv");
			receive = readCtrl();
			port = getPort(receive);

			if (port == -1) {
				rc = false;
			} else {
				if (bindDataSocket()) {
					this.setDataReader(new
NetReader(getDataSock()));
					this.setDataWriter(new
NetWriter(getDataSock()));

					setPassiveConnected();
				} else // bindDataSocket(..) FAILED
				{
					rc = false;
				}
			}
		}

		return (rc);
	}

	/**
	 * 3.4. Decompose Conditional: preparePassiveDataConnection method.
	 */
	private boolean preparePassiveDataConnection() {
		return ((isConnected()) && (!isPassiveConnected()));
	}

	/**
	 * Closes a passive connection for data.
	 *
	 * @return true if disconnected; false if not
	 *@since 0.1.0
	 *@version 0
	 */
	public boolean closePassiveDataConnection() {
		boolean rc = false;

		if (isPassiveConnected()) {
			try {
				getDataSock().close();
				setDataSock(null);
```

```
                        setDataIs(null);
                        setDataOs(null);
                        setDataIsr(null);
                        setDataReader(null);
                        setDataWriter(null);

                        rc = true;
                  } catch (IOException ioe) {
                        writeLineToConsole("E...while             closeing
dataSock.");
                  }
            }

            unsetPassiveConnected();

            return (rc);
      }

      /* <START Implementing tools for "Connectable"> */
      /**
       * Binds a CtrlSocket to the Host/Port - pair set in JFTPSuper
       *
       * @return true if socket is bind; else false
       *@since 0.1.0
       *@version 0
       */
      private boolean bindCtrlSocket() {
            return (bindSocket('c'));
      }

      /**
       *   1.12.   Remove   Parameter:   port   parameter   is   removed   from
bindDataSocket
       * method by using port instance variable.
       */

      /**
       * Binds a DataSocket to the Host/Port - pair set in JFTPSuper
       *
       * @return true if socket is bind; else false.
       *@since 0.1.0
       *@version 0
       */
      private boolean bindDataSocket() {
            return (bindSocket('d'));
      }

      /**
       * 1.12. Remove Parameter: server and port parameters are removed from
       * bindSocket method by using server and port instance variables.
       */

      /**
       * Binds a Socket to the given Host/Port - pair
       *
       * @param sock
       *            The type of socket (only 'c' or 'd' are valid).
       *@return true if socket is bind; else false
       *@since 0.1.0
       *@version 0
       */
      private boolean bindSocket(char sock) {
            boolean rc = false;
```

```
            if (getDebugLevel() >= 2)
                  printErrorLine("Binding   Socket   to   server   "   +
getServerName() + ":"
                              + getServerPort());
            /* NO! ^ Not the beer */

            try {
                  if (sock == 'd')
                        setDataSock(new            Socket(getServerName(),
getServerPort()));
                  else if (sock == 'c')
                        setCtrlSock(new            Socket(getServerName(),
getServerPort()));
                  else
                        printErrorLine("Ooops... You should __NEVER__ get
here! hum?");

                  rc = true;
            } catch (BindException be) {
                  printErrorLine("Can't bind socket.");
            } catch (MalformedURLException mue) {
                  printErrorLine("Malformed URL.");
            } catch (ConnectException ce) {
                  printErrorLine("Connection refused by Server.");
            } catch (NoRouteToHostException nrthe) {
                  printErrorLine("No route to host.");
            } catch (ProtocolException pe) {
                  printErrorLine("A protocol error occured.");
            } catch (UnknownHostException uhe) {
                  printErrorLine("Unable   to   resolve   IP-address   of
specified host.");
            } catch (UnknownServiceException use) {
                  printErrorLine("Used protocol/service is not known.");
            } catch (IOException ioe) {
                  printErrorLine("Unknown IO-Error.");
            }

            if (getDebugLevel() >= 2)
                  printErrorLine("Socket is bound (" + rc + ")");

            return (rc);
      }

      /**
       * 1.11. Remove Assignments to Parameters: assignments of parameter of
       * getPort method are removed by using temporary variable.
       */

      /**
       * Returns the port to use for PASV
       *
       * @param s
       *            String to parse for the Port
       *@return the port to use; or -1 if it can't resolve the port.
       *@since 0.0.2pA1
       */
      private int getPort(String s) {
            /*
             * Now get the port to connect to.
             *
             * To do this we have to parse a string like this:
             * "227 Entering Passive Mode (10,1,1,1,4,13)" where '10,1,1,1'
```

```
is the
		 * IP-Ad. and '4,13' stands for the port. ('4' are the first
8bits of an
		 * 16bit long number and '13' is the second.) In this case the
port
		 * would be 1037 (4256+13).
		 *
		 *  At  first  we  are  creating  a  substring  containing
"(10,1,1,1,4,13)" to
		 * parse and get the 5th and 6th Token, convert them to int and
		 * calculate the _REAL_ port to give it back.
		 */
		int rc = -1, port1, port2, i;
		StringTokenizer st;

		/**
		 *  1.11.1.  Temporary  variable  is  set  value  of  parameter.
1.11.2. All
		 * assignments to parameter are set to temporary variable.
		 */
		String temp = s;
		try {
			temp	=	temp.substring(temp.indexOf('('),
temp.indexOf(')'));

			st = new StringTokenizer(temp, ",", false);

			if (st.countTokens() == 6) {
				for (i = 0; i < 4; i++)
					st.nextToken();

				port1 = Integer.parseInt(st.nextToken());
				port2 = Integer.parseInt(st.nextToken());
				/*
				 * Now "calculate" them this way:
				 *
				 * 01010011 . 10101100 = 0101001110101100 ^ ^ ^ ^
8bit . 8bit =
				 * 16bit ^ ^ ^ ^ 1st port concatenation 2nd port =
_REAL_ port
				 */
				rc = (port1 * 256) + port2;
			}
		} catch (Exception e) {
			rc = -1;
		}

		return (rc);
	}

	/* <END Implementing tools for "Connectable"> */
	/* <END Implementing "Connectable"> */

	/**
	 * 1.6. Extract Method: printOutputLine, printOutput, printErrorLine,
	 * writeLineToConsole, readInputLine and readInputLine methods. 2.2.
Pull Up
	 * Method: printOutput, readInputLine and readInputPassword methods
are
	 * pulled up to FTPConnection.
	 */
	protected void printOutputLine(String message) {
		getStdout().println(message);
```

```
	}

	protected void printOutput(String message) {
		getStdout().print(message);
	}

	protected void printErrorLine(String message) {
		getStderr().println(message);
	}

	protected void writeLineToConsole(String message) {
		System.out.println(message);
	}

	protected String readInputLine() throws IOException {
		return getStdin().readLine();
	}

	protected String readInputPassword() throws IOException {
		return getStdin().readPasswd();
	}
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.io.File;
import java.net.InetAddress;
import java.util.StringTokenizer;

import jftprefactoring.flexibilityandextensibility.interfaces.Createable;
import
jftprefactoring.flexibilityandextensibility.interfaces.DirectoryChangeable;
import jftprefactoring.flexibilityandextensibility.interfaces.Listable;
import jftprefactoring.flexibilityandextensibility.interfaces.Removeable;

/**
 * 1.2.1. Extract Subclass: FTPDirectory extends FTPCmdServer and relevant
 * methods are pushed down. 1.4.2. FTPDirectory extends FTPConnection.
 */
public class FTPDirectory extends FTPConnection implements Createable,
		DirectoryChangeable, Listable, Removeable {

	/**
	 * 1.3.1. Push Down Field: currentWorkingDir is pushed down with
	 * encapsulation methods from JFTPSuper.
	 */
	private File currentWorkingDir = new File(".");

	/**
	 * 1.3.3. FTPDirectory constructors. 1.4.3. FTPDirectory constructors
invoke
	 * FTPConnection constructor.
	 */
	public FTPDirectory() {
		super();
	}
```

```
	public FTPDirectory(InetAddress server, int port) {
		super(server, port);
	}

	public FTPDirectory(File currentWorkingDir) {
		super();
		this.currentWorkingDir = currentWorkingDir;
	}

	public FTPDirectory(InetAddress server, int port, File
currentWorkingDir) {
		super(server, port);
		this.currentWorkingDir = currentWorkingDir;
	}

	/**
	 * 1.3.2. Encapsulation methods are pushed down with fields from
JFTPSuper.
	 */

	/**
	 * Returns a java.lang.String containing the current working directory
	 *
	 * @return A java.lang.String
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String getCurrentWorkingDir() {
		String rc = currentWorkingDir.getAbsolutePath();
		rc = rc.substring(0, rc.length() - 1);

		return (rc);
	}

	/**
	 * 4.1. Reverse Conditional: setCurrentWorkingDir method.
	 */

	/**
	 * Sets the current working dir to the given java.lang.String
	 *
	 * @param A
	 *            java.lang.String
	 *@return true if path is set; otherwise false
	 *@since 0.1.99t4
	 *@version 2
	 */
	/*
	 * Some infos: Java's path-format is
	 *  'DIR_SEPARATOR''PATH''DIR_SEPARATOR''.'. eg: / PATH / . . eg:
"/root/."
	 * || "C:\MICROS~1\."
	 */
	public boolean setCurrentWorkingDir(String newPath) {
		boolean rc = true;
		String cp;

		if (isAbsolutePath(newPath)) // absolute
		{

			/**
			 * 4.1. Reverse Conditional: existing absolute path
```

```
logic.
                     */
                    /*
                     * if (!(new File(newPath)).exists()) rc = false; else
                     * currentWorkingDir = new File(newPath + File.separator
+ ".");
                     */
                    if ((new File(newPath)).exists())
                           currentWorkingDir    =    new    File(newPath    +
File.separator + ".");
                    else
                           rc = false;
             } else if (isDirectoryUp(newPath)) // Dir up
             {
                    cp = getCurrentWorkingDir();
                    int lastIndex = cp.length() - 2; // cutting '\n' from
the end
                    StringTokenizer   st   =   new   StringTokenizer(newPath,
File.separator);

                    do {
                           lastIndex = cp.lastIndexOf("/", lastIndex);
                           cp = cp.substring(0, lastIndex);
                           // pop ".." to /dev/null (we just need the 'st' as
counter)
                           st.nextToken();
                    } while ((st.hasMoreTokens()));

                    cp += File.separator + ".";

                    if (!(new File(cp)).exists())
                           rc = false;
                    else
                           currentWorkingDir = new File(cp);
             } else // relative
             {
                    cp = getCurrentWorkingDir();
                    if (isRelativePath(cp))
                           cp = cp.substring(0, cp.length() - 1);

                    /**
                     *  4.1.  Reverse  Conditional:  existing  relative  path
logic.
                     */
                    /*
                     *  if  (!(new  File(cp  +  newPath  +  File.separator  +
".")).exists()) rc
                     *  =  false;  else  currentWorkingDir  =  new  File(cp  +
newPath +
                     * File.separator + ".");
                     */
                    if   ((new   File(cp   +   newPath   +   File.separator   +
".")).exists())
                           currentWorkingDir  =  new  File(cp  +  newPath  +
File.separator
                                         + ".");
                    else
                           rc = false;
             }

             return (rc);
      }
```

```
	/**
	 * 1.5. Hide Delegate: FTPTransfer is hidden by readData methods.
	 */
	protected String readData() {
		FTPTransfer transfer = new FTPTransfer(this.getServer(), this
				.getServerPort());
		return transfer.readData();
	}

	/**
	 * 1.2.2. Relevant methods are pushed down from FTPCmdServer.
	 */

	/* <START Implementing "Createable"> */

	/**
	 * Creates a directory on the server named dirName
	 *
	 * @param dirName
	 *            The name of the directory to create
	 *@return true if successfully created; otherwise false
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean createRemoteDir(String dirName) {
		boolean rc = false; // senseless; but the compiler wants it ;-)
		String receive;

		send("mkd " + dirName.trim());
		receive = readCtrl();
		printOutputLine(receive);

		if (pathIsCreated(receive)) {
			rc = true;
		} else if (pathAlreadyExists(receive)) {
			rc = false;
		}

		return (rc);
	}

	/**
	 * Creates a directory on the localhost named dirName
	 *
	 * @param dirName
	 *            The name of the directory to create
	 *@return true if successfully created; otherwise false
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean createLocalDir(String dirName) {
		File newDir = new File(getCurrentWorkingDir() + File.separator
				+ dirName.trim());

		return (newDir.mkdir());
	}

	/* <END Implementing "Createable"> */

	/* <START Implementing "Changeable"> */
	public boolean changeRemoteWorkingDir(String newDir) {
		boolean rc = false;
		String receive;
```

```
		send("cwd " + newDir);
		receive = readCtrl();
		printOutputLine(receive);

		if (fileActionIsOkay(receive))
			rc = true;

		return (rc);
	}

	public boolean changeLocalWorkingDir(String newDir) {
		return (setCurrentWorkingDir(newDir));
	}

	/* <END Implementing "Changeable"> */

	/* <START Implementing "Listable"> */

	/**
	 * Return the current working directory on the server.
	 *
	 * @return A java.lang.String containing the working directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printRemoteWorkingDir() {
		send("pwd");
		return (readCtrl());
	}

	/**
	 * Return the current working directory on the localhost.
	 *
	 * @return A java.lang.String containing the working directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printLocalWorkingDir() {
		return (getCurrentWorkingDir());
	}

	/**
	 * Gives a detailed directorylist of the active directory.
	 *
	 * @return Directorylist of the active directory.
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public String getLongDirList() {
		String rc = "";

		if (openPassiveDataConnection()) {
			send("list");
			rc += readCtrl();
			if (!fileActionIsNotTaken(rc)) {
				rc += readData();
				rc += readCtrl();
			}
		} else {
			rc = "Unable to establish a DataConnection.";
		}
```

```
		if (isPassiveConnected())
			rc += "Unable to close the DataConnection.";

		return (rc);
	}

	/**
	 * 4.1. Reverse Conditional: getShortDirList method.
	 */

	/**
	 * Gives a short directorylist of the active directory.
	 *
	 * @return Directorylist of the active directory.
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public String getShortDirList() {
		String rc = "";
		boolean con;

		if (!isPassiveConnected())
			con = openPassiveDataConnection();
		else
			con = true;

		if (con) {
			send("nlst");
			rc += readCtrl();

			/**
			 *  4.1.1.  Reverse  Conditional:  not  taken  file  action
logic.
			 */
			/*
			 * if (!fileActionIsNotTaken(rc)) { rc += readData(); rc
+=
			 * readCtrl(); } else { closePassiveDataConnection(); }
			 */
			if (fileActionIsNotTaken(rc)) {
				closePassiveDataConnection();
			} else {
				rc += readData();
				rc += readCtrl();
			}
		} else {
			rc = "Unable to establish a DataConnection.";
		}

		if (isPassiveConnected())
			rc += "Unable to close the DataConnection.";

		return (rc);
	}

	/**
	 *  Returns  a  java.lang.String  array  containing  the  directorylist  of
the
	 * active directory.
	 *
	 * @return A java.lang.String array.
	 *@since 0.1.99t2
	 *@version 1
```

```
         */
        public String[] getDirListArray() {
                int i;
                StringTokenizer  st  =  new  StringTokenizer(getShortDirList(),
"\n\r");
                String[] rc = new String[st.countTokens() - 2];
                String tmp = "";

                st.nextToken(); // pop "150 Openi..." to /dev/null

                for (i = 0; i < rc.length; i++) {
                        tmp = st.nextToken();
                        rc[i] = tmp;
                }

                st.nextToken(); // pop "226 Trans..." to /dev/null

                if (getDebugLevel() >= 2)
                        printErrorLine("rc has # entries: " + rc.length);

                return (rc);
        }

        /**
         * Returns a short directorylist of the current working-directory
         *
         * @return A java.lang.String
         *@since 0.1.99t2
         *@version 1
         */
        public String getLocalShortDirList() {
                int i;
                String rc = "";
                File fi = new File(getCurrentWorkingDir());
                String[] tmp = fi.list();

                for (i = 0; i < tmp.length; i++) {
                        rc += tmp[i] + "\n\r";
                }

                return (rc);
        }

        /**
         *   1.10.   Split   Loop:   loop   of   files   list   properties   in
getLocalLongDirList
         * method is duplicated to three separate loops as a loop per each
property
         * (Directory, Readable and Writable).
         */

        /**
         * Returns a long directorylist of the current working-directory
         *
         * @return A java.lang.String
         *@since 0.1.99.t2
         *@version 1
         */
        public String getLocalLongDirList() {
                String rc = "";
                String[] tmp;
                File[] fl;
                int i;
```

```
                File fi = new File(getCurrentWorkingDir());

                fl = fi.listFiles();
                tmp = new String[fl.length];

                /**
                 * 1.10.1. Extracted loop for Directory property of files list.
                 */
                for (i = 0; i < fl.length; i++) {
                       // is Directory?
                       if (fl[i].isDirectory())
                              tmp[i] = "d";
                       else
                              tmp[i] = "-";

                       // Size
                       // tmp[i] += " " + fl[i].length();

                       tmp[i] += " " + fl[i].getName();
                }

                /**
                 * 1.10.2. Extracted loop for Readable property of files list.
                 */
                for (i = 0; i < fl.length; i++) {
                       // is Readable ?
                       if (fl[i].canRead())
                              tmp[i] += "r";
                       else
                              tmp[i] += "-";

                       // Size
                       // tmp[i] += " " + fl[i].length();

                       tmp[i] += " " + fl[i].getName();
                }

                /**
                 * 1.10.3. Extracted loop for Readable property of files list.
                 */
                for (i = 0; i < fl.length; i++) {
                       // is Writable ?
                       if (fl[i].canWrite())
                              tmp[i] += "w";
                       else
                              tmp[i] += "-";

                       // Size
                       // tmp[i] += " " + fl[i].length();

                       tmp[i] += " " + fl[i].getName();
                }

                // String[] -> String
                for (i = 0; i < tmp.length; i++) {
                       rc += tmp[i] + "\n\r";
                }

                return (rc);
        }

        /**
         * Returns a java.lang.String arry containing the directorylist of the
```

```
local
        * working-directory.
        *
        * @return A java.lang.String array.
        *@since 0.1.99t2
        *@version 1
        */
       /*
        * I know that this is not a really good place... ;-( (tk)
        */
       public String[] getLocalDirListArray() {
              File fi = new File(getCurrentWorkingDir());

              return (fi.list());
       }

       /* <END Implementing "Listable"> */

       /* <START Implementing "Removeable"> */

       /**
        * Removes a remote Directory called dirName
        *
        * @param dirName
        *            The name of the Directory to remove
        *@return true if successfully removed; otherwise false
        *@since 0.1.99t4
        *@version 1
        */
       public boolean removeRemoteDir(String dirName) {
              boolean rc = false; // senseless; but the compiler wants it ;-)
              String receive;

              send("rmd " + dirName.trim());
              receive = readCtrl();
              printOutputLine(receive);

              if (fileActionIsOkay(receive)) {
                     rc = true;
              } else if (fileActionIsNotTaken(receive)) {
                     rc = false;
              }

              return (rc);
       }

       /**
        * Removes a local Directory called dirName
        *
        * @param dirName
        *            The name of the Directory to remove
        *@return true if successfully removed; otherwise false
        *@since 0.1.99t4
        *@version 2
        */
       public boolean removeLocalDir(String dirName) {
              String cd = getCurrentWorkingDir();
              File rmDir = new File(cd.substring(0, cd.length() - 1) +
dirName.trim());

              return (rmDir.delete());
       }
```

```
	/* <END Implementing "Removeable"> */

	/**
	 * 3.4. Decompose Conditional: pathIsCreated method.
	 */
	private boolean pathIsCreated(String response) {
		return (response.startsWith("257"));
	}

	/**
	 * 3.4. Decompose Conditional: pathAlreadyExists method.
	 */
	private boolean pathAlreadyExists(String response) {
		return (response.startsWith("521"));
	}

	/**
	 * 3.4. Decompose Conditional: fileActionIsOkay method.
	 */
	private boolean fileActionIsOkay(String response) {
		return (response.startsWith("250"));
	}

	/**
	 * 3.4. Decompose Conditional: actionIsNotTaken method.
	 */
	protected boolean fileActionIsNotTaken(String response) {
		return (response.equals("550"));
	}

	/**
	 * 3.4. Decompose Conditional: isAbsolutePath method.
	 */
	private boolean isAbsolutePath(String path) {
		return path.startsWith("/");
	}

	/**
	 * 3.4. Decompose Conditional: isDirectoryUp method.
	 */
	private boolean isDirectoryUp(String path) {
		return path.startsWith("..");
	}

	/**
	 * 3.4. Decompose Conditional: isRelativePath method.
	 */
	private boolean isRelativePath(String path) {
		return path.endsWith(".");
	}
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import java.io.DataInputStream;
import java.io.DataOutputStream;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.InetAddress;
import java.util.StringTokenizer;

import jftprefactoring.flexibilityandextensibility.interfaces.Transferable;
import jftprefactoring.flexibilityandextensibility.ui.Progressbar;

/**
 * 1.2.1. Extract Subclass: FTPTransfer extends FTPCmdServer and relevant
 * methods are pushed down. 1.4.2. FTPTransfer extends FTPConnection. 2.4.1.
 * Replace Delegation with Inheritance: FTPTransfer extends FTPDirectory.
 */
public class FTPTransfer extends FTPDirectory implements Transferable {

	/**
	 * 1.3.1. Push Down Field: currentWorkingDir is pushed down with
	 * encapsulation methods from JFTPSuper. 2.4.2. FTPTransfer inherits
local
	 * variables from FTPDirectory.
	 */
	// private File currentWorkingDir = new File(".");
	/**
	 * 1.3.3. FTPTransfer constructors. 1.4.3. FTPTransfer constructors
invoke
	 * FTPConnection constructor. 2.4.3. FTPTransfer constructors invoke
	 * FTPDirectory constructor.
	 */
	public FTPTransfer() {
		super();
	}

	public FTPTransfer(InetAddress server, int port) {
		super(server, port);
	}

	public FTPTransfer(File currentWorkingDir) {
		super(currentWorkingDir);
	}

	public    FTPTransfer(InetAddress    server,    int    port,    File
currentWorkingDir) {
		super(server, port, currentWorkingDir);
	}

	/**
	 * 1.3.2. Encapsulation methods are pushed down with fields from
JFTPSuper.
	 *   2.4.4.   FTPTransfer   inherits   encapsulation   methods   from
FTPDirectory.
	 */

	/**
	 * 1.5. Hide Delegate: FTPDirectory is hidden by getLongDirList
method.
	 * 2.4.5. getLongDirList method is removed due to replacing delegation
to
	 * inheritance.
```

```
         */
        /*
         * protected String getLongDirList() { File currentDirectory = new
         * File(this.getCurrentWorkingDir()); FTPDirectory dir = new
         *      FTPDirectory(this.getServer(),      this      .getServerPort(),
currentDirectory);
         * return dir.getLongDirList(); }
         */

        /**
         * 1.2.2. Relevant methods are pushed down from FTPCmdServer.
         */

        /* <START Implementing "Transferable"> */

        /**
         * Reads data from the FTPServer.
         *
         * @return String
         *@since 0.0.2pA4
         *@version 3
         */
        public String readCtrl() {
                String rc = "", line;
                int i = 0;

                try {
                        do {
                                line = "";

                                do {
                                        i = getCtrlStream().read();
                                        line += (char) i;
                                } while (i != 10);

                                rc += line;

                        } while ((char) line.charAt(3) != ' ');
                } catch (Exception e) {
                        printErrorLine("Error  while  reading  from  server  (in
fcs): "
                                        + e.toString());
                }

                return (rc);
        }

        /**
         * Reads a whole block of data from the ctrl-channel of the FTPServer.
         *
         * @return String
         *@since 0.0.2pA4
         *@version 2
         */
        public String readCtrlBlock() {
                String rc = "";
                int i = 0; // char read from server.

                try {
                        do {
                                i = getCtrlStream().read();
                                if (i == -1)
                                        break;
```

```
                        rc += (char) i;
                } while (true);
        } catch (IOException e) {
                printErrorLine("Error receiving Block.");
        }

        return (rc);
}

/**
 * Reads data from the FTPServer.
 *
 * @return String
 *@since 0.0.2pA4
 *@version 1
 */
public String readData() {
        String rc = "";
        int i = 0; // char read from server.

        try {
                i = getDataStream().read();

                while (i != -1) {
                        rc += (char) i;
                        i = getDataStream().read();
                        // printErrorLine("retrieving: " + (char) i);
                }
                closePassiveDataConnection();
        } catch (IOException e) {
                printErrorLine("Error receiving Block.");
        }

        return (rc);
}

/**
 * Reads a whole block of data from the data-channel of the FTPServer.
 *
 * @return String
 *@since 0.0.2pA4
 *@version 0
 */
public String readDataBlock() {
        String rc = "";

        while ("this".equals("that")) {
                try {
                        rc += getDataStream().read();
                } catch (Exception e) {
                        printErrorLine("Gotcha!");
                }
        } // YES, this does NOT work.

        return (rc);
}

/**
 * Reads a single line of data from the data-channel of the FTPServer.
 *
 * @return String
 *@since 0.0.2pA4
 *@version 0
```

```
         */
        public String readDataLine() {
                String rc = "";

                while ("this".equals("that")) {
                        try {
                                rc += getDataStream().read();
                        } catch (Exception e) {
                                printErrorLine("Gotcha!");
                        }
                } // YES, this does NOT work.

                return (rc);
        }

        /**
         * Sends a String to the FTPServer.
         *
         * @param s
         *             String to send
         *@since 0.0.2pA1
         */
        public void send(String s) {
                getCtrlWriter().write(s);
        }

        /**
         * Receives a list of files named 'localList' and tx's them to the
server.
         *
         * @param localList
         *             a java.lang.String array where to save the files
         *@return true if successfully transfered; else false
         *@since 0.1.99t2
         *@version 1
         */
        public boolean putFiles(String[] localList) {
                boolean rc = true;
                int i;

                for (i = 0; i < localList.length; i++) {
                        if (!putFile(localList[i]))
                                rc = false;
                }

                return (rc);
        }

        /**
         * Receives a list of files named 'serverList' and stores them in
         * 'localList'
         *
         * @param serverList
         *             a java.lang.String array of files to get
         *@param localList
         *             a java.lang.String array where to save the files
         *@return true if successfully transfered; else false
         *@since 0.1.99t2
         *@version 1
         */
        public boolean getFiles(String[] serverList, String[] localList) {
                boolean rc = true;
                int i;
```

```
            if (localList.length == serverList.length)
                  for (i = 0; i < localList.length; i++) {
                        if (!getFile(serverList[i], localList[i]))
                              rc = false;
                  }
            else
                  rc = false;

            return (rc);
      }

      /**
       * 1.10. Split Loop: loop in getFile method cannot be split to two
separate
       * loops because progress bar setting monitors status of writing read
bytes
       * process.
       */

      /**
       * Recieves a File named 'ServerFile' and stores it at 'LocalFile'
       *
       * @param serverFile
       *            The Filename at the Server's side
       *@param localFile
       *            The Filename where it is saved (local)
       *@return True if the file is successfully transfered; else false
       *@since 0.0.2pA1
       *@version 2
       */
      public boolean getFile(String serverFile, String localFile) {
            // We must catch errors like '5xx Permission denied...'
            boolean rc = false;
            int size = -1, i = 0;
            String serverResponse = "";
            FileOutputStream fos = null;

            size = getFileSize(serverFile);
            // printOutputLine("Filesize is " + size);// Should only be
used for
            // debug(tk)

            if (openPassiveDataConnection()) {
                  if (size > 0) {
                        /*
                         * Here we may look first if the file already
exists on the
                         * local fs, compare the size and may only
transfer the missing
                         * part(s).(tk)
                         */
                        send("retr " + serverFile);
                        // printOutputLine( "retrieving file" );//debug

                        // Checking return-value
                        serverResponse = readCtrl();
                        printOutputLine(serverResponse);

                        if (fileActionIsNotTaken(serverResponse)) {
                              printOutputLine("Permission denied.");
                        } else if (fileStatusIsOkay(serverResponse)) { //
Server is
```

```
                                    // ready to
                                    // transfer.
                                    try {
                                          Progressbar       pg       =       new
Progressbar(size);
                                          DataInputStream     dis     =     new
DataInputStream(
                                                      getDataInputStream());
                                          i = 0;

                                          fos               =               new
FileOutputStream(getCurrentWorkingDir()
                                                      + localFile);

                                          /*
                                           *  This  should  be  improved  because
single byte reading
                                           *   and   writing   is   _VERY_   slow.
Maybe, there is a kind of
                                           *    blockread    and    blockwrite
somewhere or we can connect
                                           * directly the dis and fos...?
                                           */
                                          /*    FML:    Whats    about    using
BufferedReader/Writer, too */
                                          /**
                                           * 1.10.1. One loop is used to write
read bytes and show
                                           * progress bar of writing process.
                                           */
                                          while (i < size) {
                                                fos.write((byte) dis.read());
                                                i++;
                                                if ((i % 1024) == 0) // 'hope
this will save some
                                                      // CPU-cycles
                                                      pg.set(i);
                                          }
                                          pg.set(i); // set the Value again to
get 100% even on
                                          // small files.

                                          closePassiveDataConnection();
                                          fos.close();
                                          dis.close();
                                          rc = true;
                                    } catch (Exception e) {
                                          printErrorLine("Error   while   Saving
File:"
                                                      + e.toString());
                                    }
                              }

                              // Closing socket && receiving server reply
                              try {
                                    printOutputLine(readCtrl());
                                    if (isPassiveConnected())
                                          closePassiveDataConnection();
                              } catch (Exception e) {
                                    printErrorLine("Can't   Read   from   ctrl-
channel.");
                              }
                        }
```

```
		}

		return (rc);
	}

	/**
	 * 1.10. Split Loop: loop in putFile method cannot be split to two
separate
	 * loops because progress bar setting monitors status of writing read
bytes
	 * process.
	 */

	/**
	 * Puts a File 'localFile' to the Server
	 *
	 * @param localFile
	 *            The file to transmit
	 *@return true if the file was transfered successfully; otherwise
false
	 *@since 0.0.2pA1
	 *@version 5
	 */
	public boolean putFile(String localFile) {
		// We must catch errors like '5xx Permission denied...'
		DataOutputStream dos;
		FileInputStream fis;
		int size, i;

		if (!openPassiveDataConnection()) {
			printOutputLine("Could not Open Passive Data
connection");
			return (false);
		}
		send("stor " + localFile);
		printOutputLine(readCtrl());

		try {
			dos = new DataOutputStream(getDataOutputStream());
			i = 0;
			fis = new FileInputStream(getCurrentWorkingDir() +
localFile);
			/*
			 * This also should be improved because single byte
writing is
			 * _VERY_ slow.
			 */
			size = fis.available();

			Progressbar pb = new Progressbar(size);

			/**
			 * 1.10.1. One loop is used to write read bytes and show
progress
			 * bar of writing process.
			 */
			while (i < size) {
				dos.write((byte) fis.read());
				i++;
				if ((i % 1024) == 0)
					pb.set(i);
			}
			pb.set(i);
```

```
                    closePassiveDataConnection();
                    fis.close();
                    dos.close();
                    // FML: What about reading "270 ..." ?
                    printOutputLine(readCtrl());
              } catch (Exception e) {
                    printOutputLine("Exception occured while sending data to
the server: "
                                + e.toString());
              }

              /* closePassiveDataConnection(); // FML: How often you want to
do this? */

              /* return ( false ); // FML: SURE???? */
              return (true);
       }

       /* <START Implementing tools for "Transferable"> */

       /* May someone improve this ugly code ? PLEASE */
       /**
        * Returns the size of a file named 'file'. It first tries to use the
size
        * command and falls back to a selfdetection mode by parsing the
output of a
        * 'list' command. If even this is not supported by the server it
fails.
        *
        * @param file
        *            The name of the file
        *@return The size of the given file; or -1 if it can't detect it.
        *@since 0.1.99
        *@version 3
        */
       private int getFileSize(String file) {
              /*
               * At first I'll try the size command, if it fails the method
tries to
               * determinate(?) the filesize by parsing the output of 'list'.
               */
              int rc = -1;
              String receive;

              send("size " + file);
              receive = readCtrl();

              if (!fileActionIsNotTaken(receive)) {
                    try {
                          if (fileStatusIsReplied(receive))
                                receive = receive.substring(3).trim();

                          rc = Integer.parseInt(receive);
                    } catch (Exception e) {
                          printErrorLine("Server   responses   an   invalid
filesize.");
                    }
              } else // Now me must go the hard way. :-(
              {
                    int i;
                    String list;
```

```
                    list = getLongDirList();

                    // Cutting the 1st and last line ("150 .." && "226 ..")
                    list = list.substring(list.indexOf("\n") + 1, // after
1st NL
                                list.lastIndexOf("\n", list.length() - 2)//
before last NL
                                );

                    /*
                     * Now we should have cut the first and last line. At
next we have
                     * to to replace '\n''s by ' ', so that the tokenizer
works proper.
                     * ARRG!, Then we have to replace '\r''s with ' ''s,
too.
                     */
                    list = list.replace('\n', ' ');
                    list = list.replace('\r', ' '); // Yes..

                    // Now we've got the String we want...going on.
                    StringTokenizer st = new StringTokenizer(list, " ");
                    String token;

                    boolean firstRun = true;
                    int size = 0;
                    int j = 1;

                    /*
                     * When tokenizing the first line, the needed fields are
8 & 4 and
                     * in any further run they are 9 & 5. Don't ask me
why... but tell
                     * me if you know it! ;-)
                     */
                    for (i = 0; st.hasMoreTokens(); i++) {
                           token = st.nextToken();

                           if (i == (9 - j)) {
                                  i = 0;
                                  if (firstRun) {
                                         firstRun = false;
                                         j = 0;
                                  }
                                  if (token.equals(file)) {
                                         //  Since  '(5-j)'  is  smaller  than
'(9-j)' we can assign
                                         // 'size' here.
                                         rc = size;
                                         break;
                                  }
                           }
                           if (i == (5 - j))
                                  size = Integer.parseInt(token);
                    }
             }

             return (rc);
      }

      /**
       * Returns the current InputStreamReader of the ctrl-channel.
       *
```

```
	 * @return InputStreamReader from the current ctrl-channel
	 *@since 0.0.2pA4
	 *@version 0
	 */
	private InputStreamReader getCtrlStream() {
		if (getCtrlIsr() == null) {
			this.setCtrlIsr(getCtrlReader().getStream());
		}

		return (getCtrlIsr());
	}

	/**
	 * Returns the current InputStreamReader of the data-channel.
	 *
	 * @return InputStreamReader from the current data-channel
	 *@since 0.1.0
	 *@version 0
	 */
	private InputStreamReader getDataStream() {
		if (getDataIsr() == null) {
			this.setDataIsr(getDataReader().getStream());
		}

		return (getDataIsr());
	}

	/**
	 * Returns the current InputStream of the data-channel.
	 *
	 * @return InputStream from the current data-channel
	 *@since 0.1.99
	 *@version 0
	 */
	private InputStream getDataInputStream() {
		if (getDataIs() == null) {
			this.setDataIs(getDataReader().getInputStream());
		}

		return (getDataIs());
	}

	/**
	 * Returns the current OutputStream of the data-channel.
	 *
	 * @return OutputStream from the current data-channel
	 *@since 0.1.99
	 *@version 0
	 */
	private OutputStream getDataOutputStream() {
		if (getDataOs() == null) {
			this.setDataOs(getDataWriter().getOutputStream());
		}

		return (getDataOs());
	}

	/* <END Implementing tools for "Transferable"> */
	/* <END Implementing "Transferable"> */

	/**
	 * 3.4. Decompose Conditional: fileStatusIsReplied method.
	 */
```

```
	private boolean fileStatusIsReplied(String response) {
		return (response.startsWith("213") && response.charAt(3) == '
');
	}

	/**
	 * 3.4. Decompose Conditional: fileStatusIsOkay method.
	 */
	private boolean fileStatusIsOkay(String response) {
		return (response.startsWith("150"));
	}
}

/*
 * Name:    JFTPSuper.java
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name               changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz       Creation
 * 0.0.2pA2 30.04.2001 Tobias Kranz       added methods to handle variables/
 *                                        options
 *   0.0.2pA4   15.05.2001   Tobias   Kranz                added   option   for
'PassiveConnected'
 *   0.0.2pA4   28.05.2001   Tobias   Kranz               added   debuginfo   for
'setServerTo(..)'
 * 0.0.2pA4 05.06.2001 Tobias Kranz       added 'getLicense()' &
 *                                        'getCopyright()' method's
 * 0.0.2pA4 06.06.2001 Tobias Kranz       removed debuginfo for
 *                                        'setServerTo(..)'
 * 0.0.2pA5 07.06.2001 Sebastian Schipper fixed serious bug in status-
 *                                        methods
 * 0.1.99t4 20.07.2001 Tobias Kranz       added 'getCurrentWorkingDir()' &&
 *                                        'setCurrentWorkingDir(String)' &&
 *                                        'String getDirSeparator()'
 * 0.1.99t5 23.07.2001 Tobias Kranz       fixed 'setCurrentWorkingDir()' &&
 *                                        removed 'getDirSeparator()' (see
 *                                        JavaDoc:java.io.File for details)
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: core package.
 */
package jftprefactoring.flexibilityandextensibility.core;

import jftprefactoring.flexibilityandextensibility.io.StdOut;

/**
 * Parameter & Object class.
 *
 * @since 0.0.2pA1
 */
public class JFTPSuper {

	/**
	 * 1.2.3.1. Remove static variable jftpSuper and remove inheritance
	 * relationships to JFTPSuper which only rely on jftpSuper.
	 */
	/**/
	// protected static JFTPSuper jftpSuper = new JFTPSuper();
```

```
	/**/

	private final String buildDate = "01.08.2001";
	private final String version = "0.1.99test6";

	/**
	 * 1.3.1. Push Down Field: connected and passiveConnected options are
pushed
	 * down with encapsulation methods to FTPConnection.
	 */
	// private BitSet options = new BitSet(4);
	/**
	 * 1.3.1. Push Down Field: server and port are pushed down with
	 * encapsulation methods to FTPConnection.
	 */
	// private InetAddress server;
	// private int port = 21;
	/**
	 * 1.3.2. Push Down Field: userName, anonymousPasswd and authenticated
are
	 * pushed down with encapsulation methods to FTPAuthentication.
	 */
	// private String userName = "me";
	// private String anonymousPasswd = "tux@northpole.org";
	// private boolean authenticated = false;
	/**
	 * 1.3.1. Push Down Field: currentWorkingDir is pushed down with
	 * encapsulation methods to FTPDirectory and FTPTransfer.
	 */
	// private File currentWorkingDir = new File(".");
	private short debugLevel = 0;
	private String info = "JavaFTP-Client v. " + version
			+ "                                                Build: " +
buildDate;

	public JFTPSuper() {
	}

	/**
	 * Returns the debugLevel
	 *
	 * @return The current debugLevel
	 *@since 0.0.2pA1
	 */
	public short getDebugLevel() {
		return ((short) debugLevel);
	}

	/**
	 * Sets the debugLevel to the short-value of 's'
	 *
	 * @param s
	 *            New debugLevel
	 *@since 0.0.2pA1
	 */
	public void setDebugLevel(short s) {
		debugLevel = s;
	}

	/**
	 * Returns a 80 character long info string.
	 *
	 * @return Info 'bout JFTP
```

```
         */
        public String getJFTPInfo() {
               return (info);
        }

        /**
         * Returns a very long info string.
         *
         * @return Info 'bout JFTP
         */
        protected String getLongJFTPInfo() {
               return ("\n"
                             + info
                             + "\n\n(c) 2000 - 2001 The JavaFTP-Group:\n\n"
                             + "    Uwe         Busch\n"
                             + "    Alexander   Flick\n"
                             + "    Christian   Kaminski\n"
                             + "    Tobias      Kranz       (TOBx@GMX.DE)\n"
                             + "    Martin      Loh         (MLoh80@GMX.DE)\n"
                             + "    Sebastian   Schipper    (dpi209@GMX.DE)\n"
                             + "    Mathias     Sroke\n"
                             +   "              Dolapo                    Falola
(Dr_Steelface@HOTMAIL.COM)\n"
                             + "    Svenja      Wittstadt\n"
                             + "\n"
                             + "This software is distributed under the rules of
the General Public " + "License(GPL) version 2 or any later version.\n");
        }

        /**
         * Returns the License valid for this application
         *
         * @return A String containing the used License
         *@since 0.0.2pA4
         *@version 1
         */
        public String getLicense() {
               return (" GNU GENERAL PUBLIC LICENSE\n Version 2, June 1991\n
\n Copyright (C) 1989, 1991 Free Software Foundation, Inc.\n 675 Mass Ave,
Cambridge, MA 02139, USA\n Everyone is permitted to copy and distribute
verbatim copies\n of this license document, but changing it is not
allowed.\n \n Preamble\n \n The licenses for most software are designed to
take away your\n freedom to share and change it.  By contrast, the GNU
General Public\n License is intended to guarantee your freedom to share and
change free\n software--to make sure the software is free for all its users.
This\n General Public License applies to most of the Free Software\n
Foundation's software and to any other program whose authors commit to\n
using it.  (Some other Free Software Foundation software is covered by\n the
GNU Library General Public License instead.)  You can apply it to\n your
programs, too.\n \n When we speak of free software, we are referring to
freedom, not\n price.  Our General Public Licenses are designed to make sure
that you\n have the freedom to distribute copies of free software (and
charge for\n this service if you wish), that you receive source code or can
get it\n if you want it, that you can change the software or use pieces of
it\n in new free programs; and that you know you can do these things.\n \n
To protect your rights, we need to make restrictions that forbid\n anyone to
deny you these rights or to ask you to surrender the rights.\n These
restrictions translate to certain responsibilities for you if you\n
distribute copies of the software, or if you modify it.\n \n For example, if
you distribute copies of such a program, whether\n gratis or for a fee, you
must give the recipients all the rights that\n you have.  You must make sure
that they, too, receive or can get the\n source code.  And you must show
them these terms so they know their\n rights.\n \n We protect your rights
```

```
with two steps: (1) copyright the software, and\n (2) offer you this license which gives you legal permission to copy,\n distribute and/or modify the software.\n \n Also, for each author's protection and ours, we want to make certain\n that everyone understands that there is no warranty for this free\n software.  If the software is modified by someone else and passed on, we\n want its recipients to know that what they have is not the original, so\n that any problems introduced by others will not reflect on the original\n authors' reputations.\n \n Finally, any free program is threatened constantly by software\n patents.  We wish to avoid the danger that redistributors of a free\n program will individually obtain patent licenses, in effect making the\n program proprietary.  To prevent this, we have made it clear that any\n patent must be licensed for everyone's free use or not licensed at all.\n \n The precise terms and conditions for copying, distribution and\n modification follow.\n \n GNU GENERAL PUBLIC LICENSE\n TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION\n \n 0. This License applies to any program or other work which contains\n a notice placed by the copyright holder saying it may be distributed\n under the terms of this General Public License.  The \"Program\", below,\n refers to any such program or work, and a \"work based on the Program\"\n means either the Program or any derivative work under copyright law:\n that is to say, a work containing the Program or a portion of it,\n either verbatim or with modifications and/or translated into another\n language.  (Hereinafter, translation is included without limitation in\n the term \"modification\".) Each licensee is addressed as \"you\".\n \n Activities other than copying, distribution and modification are not\n covered by this License; they are outside its scope.  The act of\n running the Program is not restricted, and the output from the Program\n is covered only if its contents constitute a work based on the\n Program (independent of having been made by running the Program).\n Whether that is true depends on what the Program does.\n \n 1. You may copy and distribute verbatim copies of the Program's\n source code as you receive it, in any medium, provided that you\n conspicuously and appropriately publish on each copy an appropriate\n copyright notice and disclaimer of warranty; keep intact all the\n notices that refer to this License and to the absence of any warranty;\n and give any other recipients of the Program a copy of this License\n along with the Program.\n \n You may charge a fee for the physical act of transferring a copy, and\n you may at your option offer warranty protection in exchange for a fee.\n \n 2. You may modify your copy or copies of the Program or any portion\n of it, thus forming a work based on the Program, and copy and\n distribute such modifications or work under the terms of Section 1\n above, provided that you also meet all of these conditions:\n \n a) You must cause the modified files to carry prominent notices\n stating that you changed the files and the date of any change.\n \n b) You must cause any work that you distribute or publish, that in\n \nwhole or in part contains or is derived from the Program or any part\n thereof, to be licensed as a whole at no charge to all third\n parties under the terms of this License.\n \n c) If the modified program normally reads commands interactively\n when run, you must cause it, when started running for such\n interactive use in the most ordinary way, to print or display an\n announcement including an appropriate copyright notice and a\n notice that there is no warranty (or else, saying that you provide\n a warranty) and that users may redistribute the program under\n these conditions, and telling the user how to view a copy of this\n License.  (Exception: if the Program itself is interactive but\n does not normally print such an announcement, your work based on\n the Program is not required to print an announcement.)\n \n These requirements apply to the modified work as a whole.  If\n identifiable sections of that work are not derived from the Program,\n and can be reasonably considered independent and separate works in\n themselves, then this License, and its terms, do not apply to those\n sections when you distribute them as separate works.  But when you\n distribute the same sections as part of a whole which is a work based\n on the Program, the distribution of the whole must be on the terms of\n this License, whose permissions for other licensees extend to the\n entire whole, and thus to each and every part regardless of who wrote it.\n
```

\n Thus, it is not the intent of this section to claim rights or contest\n your rights to work written entirely by you; rather, the intent is to\n exercise the right to control the distribution of derivative or\n collective works based on the Program.\n \n In addition, mere aggregation of another work not based on the Program\n with the Program (or with a work based on the Program) on a volume of\n a storage or distribution medium does not bring the other work under\n the scope of this License.\n \n 3. You may copy and distribute the Program (or a work based on it,\n under Section 2) in object code or executable form under the terms of\n Sections 1 and 2 above provided that you also do one of the following:\n \n a) Accompany it with the complete corresponding machine-readable\n source code, which must be distributed under the terms of Sections\n 1 and 2 above on a medium customarily used for software interchange; or,\n \n b) Accompany it with a written offer, valid for at least three\n years, to give any third party, for a charge no more than your\n cost of physically performing source distribution, a complete\n machine-readable copy of the corresponding source code, to be\n distributed under the terms of Sections 1 and 2 above on a medium\n customarily used for software interchange; or,\n \n c) Accompany it with the information you received as to the offer\n to distribute corresponding source code. (This alternative is\n allowed only for noncommercial distribution and only if you\n received the program in object code or executable form with such\n an offer, in accord with Subsection b above.)\n \n The source code for a work means the preferred form of the work for\n making modifications to it. For an executable work, complete source\n code means all the source code for all modules it contains, plus any\n associated interface definition files, plus the scripts used to\n control compilation and installation of the executable. However, as a\n special exception, the source code distributed need not include\n anything that is normally distributed (in either source or binary\n form) with the major components (compiler, kernel, and so on) of the\n operating system on which the executable runs, unless that component\n itself accompanies the executable.\n \n If distribution of executable or object code is made by offering\n access to copy from a designated place, then offering equivalent\n access to copy the source code from the same place counts as\n distribution of the source code, even though third parties are not\n compelled to copy the source along with the object code.\n \n 4. You may not copy, modify, sublicense, or distribute the Program\n except as expressly provided under this License. Any attempt\n otherwise to copy, modify, sublicense or distribute the Program is\n void, and will automatically terminate your rights under this License.\n However, parties who have received copies, or rights, from you under\n this License will not have their licenses terminated so long as such\n parties remain in full compliance.\n \n 5. You are not required to accept this License, since you have not\n signed it. However, nothing else grants you permission to modify or\n distribute the Program or its derivative works. These actions are\n prohibited by law if you do not accept this License. Therefore, by\n modifying or distributing the Program (or any work based on the\n Program), you indicate your acceptance of this License to do so, and\n all its terms and conditions for copying, distributing or modifying\n the Program or works based on it.\n \n 6. Each time you redistribute the Program (or any work based on the\n Program), the recipient automatically receives a license from the\n original licensor to copy, distribute or modify the Program subject to\n these terms and conditions. You may not impose any further\n restrictions on the recipients' exercise of the rights granted herein.\n You are not responsible for enforcing compliance by third parties to\n \nthis License. \n \n7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues),\n conditions are imposed on you (whether by court order, agreement or\n otherwise) that contradict the conditions of this License, they do not\n excuse you from the conditions of this License. If you cannot\n distribute so as to satisfy simultaneously your obligations under this\n License and any other pertinent obligations, then as a consequence you\n may not distribute the Program at all. For example, if a patent\n license would

```
not permit royalty-free redistribution of the Program by\n all those who receive copies directly or indirectly through you, then\n the only way you could satisfy both it and this License would be to\n refrain entirely from distribution of the Program.\n \n If any portion of this section is held invalid or unenforceable under\n any particular circumstance, the balance of the section is intended to\n apply and the section as a whole is intended to apply in other\n circumstances.\n \n It is not the purpose of this section to induce you to infringe any\n patents or other property right claims or to contest validity of any\n such claims; this section has the sole purpose of protecting the\n integrity of the free software distribution system, which is\n implemented by public license practices.  Many people have made\n generous contributions to the wide range of software distributed\n through that system in reliance on consistent application of that\n system; it is up to the author/donor to decide if he or she is willing\n to distribute software through any other system and a licensee cannot\n impose that choice.\n \n This section is intended to make thoroughly clear what is believed to\n be a consequence of the rest of this License.\n \n 8. If the distribution and/or use of the Program is restricted in\n certain countries either by patents or by copyrighted interfaces, the\n original copyright holder who places the Program under this License\n may add an explicit geographical distribution limitation excluding\n those countries, so that distribution is permitted only in or among\n countries not thus excluded.  In such case, this License incorporates\n the limitation as if written in the body of this License.\n \n 9. The Free Software Foundation may publish revised and/or new versions\n of the General Public License from time to time.  Such new versions will\n be similar in spirit to the present version, but may differ in detail to\n address new problems or concerns.\n \n Each version is given a distinguishing version number.  If the Program\n specifies a version number of this License which applies to it and \"any\n later version\", you have the option of following the terms and conditions\n either of that version or of any later version published by the Free\n Software Foundation.  If the Program does not specify a version number of\n this License, you may choose any version ever published by the Free Software\n Foundation.\n \n 10. If you wish to incorporate parts of the Program into other free\n programs whose distribution conditions are different, write to the author\n to ask for permission.  For software which is copyrighted by the Free\n Software Foundation, write to the Free Software Foundation; we sometimes\n make exceptions for this.  Our decision will be guided by the two goals\n of preserving the free status of all derivatives of our free software and\n of promoting the sharing and reuse of software generally.\n \n NO WARRANTY\n \n 11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY\n FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN\n OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES\n PROVIDE THE PROGRAM \"AS IS\" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED\n OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF\n MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS\n TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE\n PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,\n REPAIR OR CORRECTION.\n \n 12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING\n WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR\n REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,\n INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING\n OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED\n TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY\n YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER\n PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE\n POSSIBILITY OF SUCH DAMAGES.\n \n END OF TERMS AND CONDITIONS");
    }

    /**
     * Returns a short information about the License
     *
```

```
        * @return A String containing a short information about the used
license
        *@since 0.0.2pA4
        *@version 0
        */
       public String getLicenseInfo() {
              return ("This Software comes without any warranty.\n"
                           + "You may redistribute and/or modify it under the
terms of the GPL v 2.\n"
                           + "For details type 'license'.\n");
       }

       /**
        * Returns the copyright informations valid for this software
        *
        * @return A String containing copyright informations
        *@since 0.0.2pA4
        *@version 0
        */
       public String getCopyright() {
              return ("Copyright 2000 - 2001 The JavaFTP-Group.\n"
                           + "For details type 'info'.\n");
       }

       /**
        * Easter egg
        *
        * @since 0.0.2pA4
        *@version [.,xX@Xx,.]
        */
       public void printLongJFTPInfo() {
              final boolean z = false;
              int i = ((!!!!!z) ? (-1) : (0));
              try {
                     for (;;) {
                           Thread.sleep(((!!!!!z) ? (0) : (1)), ((!!!!z) ?
(0) : (1)));
                           (new
StdOut()).print(getLongJFTPInfo().charAt(++i));
                           if (i >= getLongJFTPInfo().length())
                                  break;
                     }
              } catch (Exception e) {
              }
       }
}

/*
 * Name:    Authenticateable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 17.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 06.06.2001 Tobias Kranz Changed name to __REAL__ english ;-)
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
```

```
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * Defines methods to authenticate a user.
 *
 * @since 0.0.2pA1
 */
/**
 *    1.9.1.    Inline    Class:    JFTPSuperInterface    is    in-lined    into
Authenticateable.
 * 1.9.2. Inheritance relationship is removed because it is not used.
 */
public interface Authenticateable {
       /**
        * Authenticates an user.
        *
        * @return true if authentication succeds; otherwise false
        *@since 0.0.2pA1
        */
       public boolean authenticate();
}

/*
 * Name:    Connectable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * Defines methods to connect to a server.
 *
 * @since 0.0.2pA1
 */
/**
 *  1.9.1.  Inline  Class:  JFTPSuperInterface  is  in-lined  into  Connectable.
1.9.2.
 * Inheritance relationship is removed it is not used.
 */
public interface Connectable {
       public boolean connect();

       public boolean disconnect();

       /**
        * Opens a ctrl connection.
        *
        * @since 0.0.2pA4
        */
       public boolean openCtrlConnection();

       /**
        * Closes a ctrl connection.
```

```
         *
         * @since 0.0.2pA4
         */
        public boolean closeCtrlConnection();

        /**
         * Opens a passive data connection.
         *
         * @since 0.0.2pA4
         */
        public boolean openPassiveDataConnection();

        /**
         * Closes a passive data connection.
         *
         * @since 0.0.2pA4
         */
        public boolean closePassiveDataConnection();
}

/*
 * Name:    Createable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 19.07.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * Defines methods to create directories and files.
 *
 * @since 0.1.99t4
 */
/**
 *  1.9.1.  Inline  Class:  JFTPSuperInterface  is  in-lined  into  Createable.
1.9.2.
 * Inheritance relationship is removed it is not used.
 */
public interface Createable {
        /**
         * Creates a Directory on the server with the given name
         *
         * @return True if successfully created; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean createRemoteDir(String dirName);

        /**
         * Creates a Directory on the local fs with the given name
         *
         * @return True if successfully created; otherwise false.
         *@since 0.1.99t4
         *@version 1
```

```
         */
        public boolean createLocalDir(String dirName);
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * 1.9.1. Inline Class: JFTPSuperInterface and Changeable are in-lined into
 * DataTypeChangeable. 1.9.2. Inheritance relationship is removed it is not
 * used.
 */
/**
 *   1.2.1.   Extract   Subclass   (Interface):   DataTypeChangeable   extends
Changeable
 * and setTxMode methods are pushed down.
 */
public interface DataTypeChangeable {

        /**
         *   1.2.2.   changeRemoteWorkingDir   method   is   pushed   down   from
Changeable.
         */
        /**
         * Sets the Tx mode either to "ascii" or to "binary"
         *
         * @param A
         *             java.lang.String containing the wished Tx-mode (Either
         *             'binary' or 'ascii')
         *@return true if succeeded; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean setTxMode(String nm);

        /**
         *   1.2.2.   changeRemoteWorkingDir   method   is   pushed   down   from
Changeable.
         */
        /**
         * Sets the Tx mode either to "ascii" or to "binary"
         *
         * @param A
         *              char containing the wished Tx-mode (Either 'i'/'b' for
binary
         *             or 'a' for ascii)
         *@return true if succeeded; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean setTxMode(char nm);
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
```

```
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * 1.9.1. Inline Class: JFTPSuperInterface and DirectoryChangeable are in-
lined into
 * DataTypeChangeable. 1.9.2. Inheritance relationship is removed it is not
 * used.
 */
/**
 *  1.2.1. Extract Subclass (Interface): DirectoryChangeable extends
Changeable
 * and changeRemoteWorkingDir and changeRemoteWorkingDir methods are pushed
 * down.
 */

public interface DirectoryChangeable {

       /**
        *  1.2.2. changeRemoteWorkingDir method is pushed down from
Changeable.
        */
       /**
        * Changes the remote working dir.
        *
        * @param A
        *            java.lang.String containing the new path.
        *@return true if succeeded; else false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean changeRemoteWorkingDir(String newPath);

       /**
        * 1.2.2. changeLocalWorkingDir method is pushed down from Changeable.
        */
       /**
        * Changes the local working dir.
        *
        * @param A
        *            java.lang.String containing the new path.
        *@return true if succeeded; else false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean changeLocalWorkingDir(String newPath);
}

/*
 * Name:    Listable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 06.05.2001 Tobias Kranz Rectifing comments
 * 0.1.99t2 17.07.2001 Tobias Kranz Added 4 methods.
 * 0.1.99t4 19.07.2001 Tobias Kranz Added 2 ~'printWorkingDir' methods.
 */
/**
 * Refactoring JFTP to Framework
```

```
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * Defines methods to list directories.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.9.1. Inline Class: JFTPSuperInterface is in-lined into Listable. 1.9.2.
 * Inheritance relationship is removed it is not used.
 */
public interface Listable {
	/**
	 * Prints the current working directory on the server.
	 *
	 * @return A java.lang.String containing the directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printRemoteWorkingDir();

	/**
	 * Prints the current working directory on the local host.
	 *
	 * @return A java.lang.String containing the directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printLocalWorkingDir();

	/**
	 * Returns a detailed directory listing of the active directory.
	 *
	 * @return Directorylist of the active directory
	 *@since 0.0.2pA1
	 */
	public String getLongDirList();

	/**
	 * Returns a short directory listing of the active directory.
	 *
	 * @return Directorylist of the active directory
	 *@since 0.0.2pA1
	 */
	public String getShortDirList();

	/**
	 * Returns a short directory listing of the active directory in a
	 * jav.lang.String array.
	 *
	 * @return A java.lang.String array containing the Directorylist of
the
	 *         active directory
	 *@since 0.1.99t2
	 *@version 1
	 */
	public String[] getDirListArray();
```

```
	/**
	 *  Returns a detailed directory listing of the active working-
directory.
	 *
	 * @return Directorylist of the active working-directory
	 *@since 0.0.2pA1
	 */
	public String getLocalLongDirList();

	/**
	 * Returns a short directory listing of the active working-directory.
	 *
	 * @return Directorylist of the active working-directory
	 *@since 0.0.2pA1
	 */
	public String getLocalShortDirList();

	/**
	 * Returns a short directory listing of the active working-directory
in a
	 * jav.lang.String array.
	 *
	 * @return A java.lang.String array containing the Directorylist of
the
	 *         active working-directory
	 *@since 0.1.99t2
	 *@version 1
	 */
	public String[] getLocalDirListArray();
}

/*
 * Name:    Removeable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 19.07.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * Defines methods to remove directories and files.
 *
 * @since 0.1.99t4
 */
/**
 *  1.9.1.  Inline  Class:  JFTPSuperInterface  is  in-lined  into  Removeable.
1.9.2.
 * Inheritance relationship is removed it is not used.
 */
public interface Removeable {
	/**
	 * Removes a Directory on the server with the given name
	 *
	 * @return True if successfully removed; otherwise false.
```

```
         *@since 0.1.99t4
         *@version 1
         */
        public boolean removeRemoteDir(String dirName);

        /**
         * Removes a Directory on the local fs with the given name
         *
         * @return True if successfully removed; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean removeLocalDir(String dirName);
}

/*
 * Name:    Transferable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 *  0.0.2pA2  03.05.2001  Tobias  Kranz  Changed  'receive-Line/-Block'  to
'receive()'
 * 0.1.0    09.06.2001 Martin Loh   Changed the 'getFile()' and 'putFile()'
 *                                          method definitions so that they
 *                                                return an boolean and have
Strings
 *                                          as args.
 * 0.1.99t2 17.07.2001 Tobias Kranz Added 'getFiles(String[], String[])',
 *                                  corrected some comments.
 * 0.1.99t4 19.07.2001 Tobias Kranz cleaned up
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: interfaces package.
 */
package jftprefactoring.flexibilityandextensibility.interfaces;

/**
 * Defines methods to transfer "raw" data and files.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.9.1. Inline Class: JFTPSuperInterface is in-lined into Transferable.
1.9.2.
 * Inheritance relationship is removed it is not used.
 */
public interface Transferable {
        /**
         * Reads data from the ctrl-channel.
         *
         * @return The data read.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readCtrl();

        /**
```

```
         * Receives a whole block of data.
         *
         * @return The block received
         *@since 0.0.2pA4
         *@version 0
         */
        public String readCtrlBlock();

        /**
         * Reads data until end of Stream
         *
         * @return The data read.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readData();

        /**
         * Reads a whole block of data.
         *
         * @return The block received
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataBlock();

        /**
         * Reads a single line of data.
         *
         * @return The data received.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataLine();

        /**
         * Recieves a list of Files 'serverlist' and saves it to 'localList'.
         *
         * @param serverList
         *            Filelist (java.lang.String array)
         *@param localList
         *            Filelist (java.lang.String array)
         *@return true if successfully transfered; otherwise false.
         *@since 0.1.99t2
         *@version 1
         */
        public boolean getFiles(String[] serverList, String[] localList);

        /**
         * Recieves a File with name 'Serverfile' and saves it to 'localfile'.
         *
         * @param ServerFile
         *            Filename
         *@param LocalFile
         *            Filename
         *@return true if successfully transfered; otherwise false.
         *@since 0.0.2pA1
         *@version 1
         */
        public boolean getFile(String ServerFile, String LocalFile);

        /**
         * Puts a File named 'LocalFile'.
```

```
         *
         * @param LocalFile
         *            Filename
         *@return true if the file was transfered successfully; false if not.
         *@since 0.0.2pA1
         *@version 0
         */
        public boolean putFile(String LocalFile);

        /**
         * Sends a string of data.
         *
         * @param s
         *            String to send.
         *@since 0.0.2pA1
         *@version 0
         */
        public void send(String s);
}

/*
 * Name:    NetIO.java (extends JFTPIO)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: io package.
 */
package jftprefactoring.flexibilityandextensibility.io;

/**
 * Class for network-io.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.9.1. Inline Class: JFTPIO is in-lined into NetIO and inheritance
 * relationship is directly moved to NetIO. 1.9.2. Inheritance relationship
is
 * removed due to its limited usage.
 */
public class NetIO {
        /**
         * 2.1. Pull Up Field: stdout and stderr are pulled up to NetIO.
         */
        private StdOut stdout = new StdOut();
        private StdErr stderr = new StdErr();

        /**
         * 1.1. Encapsulate Fields: getStdout and setStdout. 2.2. Pull Up
Method:
         * getStdout and setStdout methods are pulled up to NetIO.
         */

        /**
         * @return the stdout
```

```
         */
        protected StdOut getStdout() {
                return stdout;
        }

        /**
         * @param stdout
         *            the stdout to set
         */
        protected void setStdout(StdOut stdout) {
                this.stdout = stdout;
        }

        /**
         * 1.1. Encapsulate Fields: getStdout and setStdout. 2.2. Pull Up
Method:
         * getStdout and setStdout methods are pulled up to NetIO.
         */

        /**
         * @return the stderr
         */
        protected StdErr getStderr() {
                return stderr;
        }

        /**
         * @param stderr
         *            the stderr to set
         */
        protected void setStderr(StdErr stderr) {
                this.stderr = stderr;
        }
}

/*
 *  Name:         NetReader.java  (extends  NetIO  (extends  JFTPIO  (extends
JFTPSuper)))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.1    12.03.2001 Tobias Kranz creation.
 *          13.03.2001 Tobias Kranz changed from Thread to regular class
 *                                  (nearly total rewrite).
 * 0.0.1pA3 15.03.2001 Tobias Kranz splitted from JFTP; nothing big.
 * 0.0.1pA6 03.04.2001 Tobias Kranz added comments.
 * 0.0.1pA7 04.04.2001 Tobias Kranz splitted 'read()' to readLine() and
 *                                  readBlock().
 * 0.0.2pA2 03.05.2001 Tobias Kranz (re)added 'read()' which implements now
 *                                  'readLine()' & 'readBlock()'
 *   0.0.2pA4   15.05.2001   Tobias   Kranz   removed   everything!   created
'readStream()'
 *                                   which must be wrapped by 'receiveLine()'
&&
 *                                  'receiveBlock()' in FTPCmdServer.java .
 *                                  See README.DEVELOPMENT for details.
 * 0.1.99   10.06.2001 Tobias Kranz added 'getInputStream()'
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
```

```
/**
 * 6.1. Extract Package: io package.
 */
package jftprefactoring.flexibilityandextensibility.io;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Socket;

/**
 * Class NetReader may be used to read any kind of data through an existing
 * Socket.
 *
 * @since 0.0.1
 */
public class NetReader extends NetIO {
	private InputStream inStream;

	/**
	 * Sets the InputStream.
	 *
	 * @param socket
	 *               Socket to listen on (This does NOT mean a listening
socket!).
	 *@since 0.0.1
	 */
	public NetReader(Socket socket) {
		try {
			this.inStream = socket.getInputStream();
		} catch (Exception e) {
			getStderr().println("Unable to get InputStream from
Socket.");
			getStdout().println("Unable to get InputStream from
Socket.");
		}
	}

	/*
	 * Get the stuff in.
	 */
	/**
	 * Autodetects the number of lines to read and give them back.
	 *
	 * @return String read from server
	 *@since 0.0.2pA2
	 */
	/*
	 * public String read() { String rc = "", line; int i;
	 *
	 * InputStreamReader isr = new InputStreamReader(inStream);
	 *
	 * try { do // start reading the whole block (or just a line..) { line
= "";
	 *
	 * do // beginn reading the line { i = isr.read();/? should we make
	 * 'isr.ready()' call(s) ? line += (char)i; } while (i != 10 ); //
until
	 * '(char)10' (means CR (0x0A)) is read
	 *
	 * rc += line; // adding the read line to the return value
	 *
	 * } while ((char)line.charAt(3) != ' '); } catch (IOException ioe)
{ /
```

```
         * shall we do something here or should we throw this Exception ?? }
         *
         *  return( rc.substring(0, (rc.length() - 1)) ); // trim last char
('\n') }
         */
        /**
         *  Returns a InputStreamReader reading from the InputStream of the
given
         * Socket.
         *
         * @return InputStreamReader
         *@since 0.0.2pA4
         */
        public InputStreamReader getStream() {
               return (new InputStreamReader(inStream));
        }

        /**
         * Returns the InputStream of the used socket
         *
         * @return An InputStream
         *@since 0.1.99
         *@version 0
         */
        public InputStream getInputStream() {
               return (inStream);
        }

        /**
         * Reads a single line of data from the server .
         *
         * @return Whatever the ftpserver sends us.
         *@since 0.0.1
         *@deprecated
         */
        /*
         * public String readLine() { return( this.read() ); } // just to stay
         * compatible
         */

        /**
         * Reads a whole block of data from the server.
         *
         * @return Whatever the server sends.
         *@since 0.0.1
         *@deprecated
         */
        /*
         * public String readBlock() { String rc = ""; boolean done = false;
int i;
         * //char read from server
         *
         * try { InputStreamReader isr = new InputStreamReader(inStream);
         *
         * while (! done) { i = isr.read();
         *
         * if (i == -1) // -1 ~ EOF { if (jftpSuper.getDebugLevel() >= 2)
         * getStdout().println("finished reading block"); done = true; } else
{ rc
         * += (char)i; if (jftpSuper.getDebugLevel() >= 2)
         * getStdout().println("Reading " + rc); } } } catch (Exception e) {
         *      getStderr().println("Exception     while     reading     from
server."+e.toString());
```

```
        * }
        *
        * return rc; }
        */
}

/*
 *  Name:        NetWriter.java  (extends  NetIO  (extends  JFTPIO  (extends
JFTPSuper)))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.1    12.03.2001 Tobias Kranz Creation
 *          13.03.2001 Tobias Kranz Changed from Thread to regular class
 *                                  (nearly total rewrite)
 * 0.0.1pA3 15.03.2001 Tobias Kranz splitted from JFTP; nothing big
 * 0.1.99   10.06.2001 Tobias Kranz added 'getOutputStream()'
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: io package.
 */
package jftprefactoring.flexibilityandextensibility.io;

import java.io.OutputStream;
import java.net.Socket;

/**
 * Class NetWriter may be used to send any kind of data through an existing
 * Socket.
 *
 * @since 0.0.1
 */

/**
 * 1.9.2. Inheritance relationship is removed due to its limited usage.
1.14.
 * Move Field: debugLevel local variable is copied from JFTPSuper to replace
 * inheritance.
 */
public class NetWriter extends NetIO {

       private OutputStream outStream;

       /**
        * 1.14. Instance variable replaces inheritance relationship.
        */
       private short debugLevel;

       /**
        * 1.14. NetWriter constructors.
        */
       public NetWriter() {
              this.debugLevel = 0;
       }

       public NetWriter(short debugLevel) {
              this.debugLevel = debugLevel;
       }
```

```
	/**
	 * Sets the OutputStream.
	 *
	 * @param socket
	 *            Socket to write through.
	 *@since 0.0.1
	 */
	public NetWriter(Socket socket) {
		this.debugLevel = 0;
		try {
			this.outStream = socket.getOutputStream();
		} catch (Exception e) {
		}
	}

	public NetWriter(Socket socket, short debugLevel) {
		this.debugLevel = debugLevel;
		try {
			this.outStream = socket.getOutputStream();
		} catch (Exception e) {
		}
	}

	/**
	 * Returns the OutputStream of the used socket
	 *
	 * @return The OutputStream used for the current socket.
	 *@since 0.1.99
	 *@version 0
	 */
	public OutputStream getOutputStream() {
		return (outStream);
	}

	/*
	 * Just send everything out to the server
	 */
	/**
	 * Writes data to the ftpserver
	 *
	 * @param string
	 *            The data to send
	 *@since 0.0.1
	 */
	public void write(String string) {
		int i;

		try {
			if (debugLevel >= 1)
				getStdout().println("starting new write-loop");

			for (i = 0; i < string.length(); i++) {
				outStream.write(string.charAt(i));
				if (debugLevel >= 2)
					getStdout().println("writing   char:   "   +
string.charAt(i));
			}
			if (string.charAt(string.length() - 1) != '\n')
				outStream.write('\n'); // Finish the line :-)
			else if (debugLevel >= 2)
				getStderr().println("DON'T SEND NEWLINES!");
```

```
                    outStream.flush(); // make sure the chars are written

                    if (debugLevel >= 1)
                          getStdout().println("ending write-loop");
             } catch (Exception e) {
                    getStderr().println("Exception in Outp(" + e.toString()
+ ")");
             }
       }
}

/*
 * Name:    StdErr.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: io package.
 */
package jftprefactoring.flexibilityandextensibility.io;

/**
 * Represents the standard error channel (StdErr).<BR>
 * The trailing '*' will be removed in stable versions.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.9.1. Inline Class: JFTPIO and StdIO are in-lined into StdErr and
inheritance
 * relationship is directly moved to StdErr. 1.9.2. Inheritance relationship
is
 * removed because it is not used.
 */
public class StdErr {
       /**
        * Prints an object to StdErr.
        *
        * @param o
        *            Object to print
        *@since 0.0.2pA1
        */
       public void print(Object o) {
             System.err.print("*" + o);
       }

       /**
        * Prints a double to StdErr.
        *
        * @param d
        *            double to print
        *@since 0.0.2pA1
        */
       public void print(double d) {
             System.err.print("*" + d);
       }
```

```
	/**
	 * Prints a float to StdErr.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void print(float f) {
		System.err.print("*" + f);
	}

	/**
	 * Prints a long to StdErr.
	 *
	 * @param l
	 *            long to print
	 *@since 0.0.2pA1
	 */
	public void print(long l) {
		System.err.print("*" + l);
	}

	/**
	 * Prints an int to StdErr.
	 *
	 * @param i
	 *            int to print
	 *@since 0.0.2pA1
	 */
	public void print(int i) {
		System.err.print("*" + i);
	}

	/**
	 * Prints a short to StdErr.
	 *
	 * @param s
	 *            short to print
	 *@since 0.0.2pA1
	 */
	public void print(short s) {
		System.err.print("*" + s);
	}

	/**
	 * Prints a char to StdErr.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void print(char c) {
		System.err.print("*" + c);
	}

	/**
	 * Prints a byte to StdErr.
	 *
	 * @param b
	 *            byte to print
	 *@since 0.0.2pA1
	 */
```

```
	public void print(byte b) {
		System.err.print("*" + b);
	}

	/**
	 * Prints a boolean to StdErr.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void print(boolean b) {
		System.err.print("*" + b);
	}

	/**
	 * Prints an object and "\n\r" to StdErr.
	 *
	 * @param o
	 *            Object to print
	 *@since 0.0.2pA1
	 */
	public void println(Object o) {
		System.err.println("*" + o);
	}

	/**
	 * Prints a double and "\n\r" to StdErr.
	 *
	 * @param d
	 *            double to print
	 *@since 0.0.2pA1
	 */
	public void println(double d) {
		System.err.println("*" + d);
	}

	/**
	 * Prints a float and "\n\r" to StdErr.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void println(float f) {
		System.err.println("*" + f);
	}

	/**
	 * Prints a long and "\n\r" to StdErr.
	 *
	 * @param l
	 *            long to print
	 *@since 0.0.2pA1
	 */
	public void println(long l) {
		System.err.println("*" + l);
	}

	/**
	 * Prints a int and "\n\r" to StdErr.
	 *
	 * @param i
```

```
         *            int to print
         *@since 0.0.2pA1
         */
        public void println(int i) {
               System.err.println("*" + i);
        }

        /**
         * Prints a short and "\n\r" to StdErr.
         *
         * @param s
         *            short to print
         *@since 0.0.2pA1
         */
        public void println(short s) {
               System.err.println("*" + s);
        }

        /**
         * Prints a char and "\n\r" to StdErr.
         *
         * @param c
         *            char to print
         *@since 0.0.2pA1
         */
        public void println(char c) {
               System.err.println("*" + c);
        }

        /**
         * Prints a byte and "\n\r" to StdErr.
         *
         * @param b
         *            byte to print
         *@since 0.0.2pA1
         */
        public void println(byte b) {
               System.err.println("*" + b);
        }

        /**
         * Prints a boolean and "\n\r" to StdErr.
         *
         * @param b
         *            boolean to print
         *@since 0.0.2pA1
         */
        public void println(boolean b) {
               System.err.println("*" + b);
        }
}

/*
 * Name:    StdIn.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.1.99t5 01.08.2001 Tobias Kranz added 'readPassword()'
 */
/**
 * Refactoring JFTP to Framework
```

```
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: io package.
 */
package jftprefactoring.flexibilityandextensibility.io;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * Represents the standard input channel (StdIn).
 *
 * @since 0.0.2pA1
 */
/**
 *  1.9.1.  Inline  Class:  JFTPIO  and  StdIO  are  in-lined  into  StdIn  and
inheritance
 * relationship is directly moved to StdIn. 1.9.2. Inheritance relationship
is
 * removed because it is not used.
 */
public class StdIn {
	private BufferedReader br;

	public StdIn() {
		this.br = new BufferedReader(new InputStreamReader(System.in));
	}

	/**
	 * Reads a whole line of data from the standard input channel.
	 *
	 * @return The String beeing read.
	 *@throws IOException
	 *             if reading fails.
	 *@since 0.0.2pA1
	 */
	public String readLine() throws IOException {
		return ((String) br.readLine());
	}

	/**
	 * Reads a single char of data from the standard input channel.
	 *
	 * @return The char beeing read.
	 *@throws IOException
	 *             if reading fails.
	 *@since 0.0.2pA1
	 */
	protected char readChar() throws IOException {
		return ((char) br.read());
	}

	/**
	 * Reads an int from the standard input channel.
	 *
	 * @return The int being read
	 *@throws IOException
	 *             if reading fails
	 *@since 0.0.2pA1
```

```
         */
        protected int readInt() throws IOException {
                return (Integer.valueOf(br.readLine()).intValue());
        }

        /**
         * Reads an InetAddress from the standard input channel.
         *
         * @return The InetAddress read
         *@throws IOException
         *             if reading fails
         *@throws UnknownHostException
         *             if host is unknown
         *@since 0.0.2pA1
         */
        protected InetAddress readInetAddress() throws IOException,
                        UnknownHostException {
                return (InetAddress.getByName(br.readLine()));
        }

        /**
         * Reads a password and dissables the terminal-echo.
         *
         * @return The Password read from StdIn
         *@throws IOException
         *             if reading fails
         *@since 0.1.99t5
         *@version 1
         */
        public String readPasswd() throws IOException {
                String rc = "";
                StringBuffer sb = new StringBuffer();
                // char c;//////////////////////////////////////

                /*
                 * Font-Colors:
                 *
                 *  Font-Colors are from 29 - 39. Background-Colors beginning
with 40.
                 *
                 *  #  color  29  Gray  30  Black  31  DarkRed  32  DarkGreen  33
Brown/DarkOrange
                 * 34 DarkBlue 35 Purple 36 Marine 37 Gray 38 White 39 Gray
                 */

                // Set Font-Color to Black
                sb.append('\u001b');
                sb.append('[');
                sb.append(30); // Color
                sb.append('m');
                System.out.print(sb.toString()); // Set it.

                // read
                rc = br.readLine();

                // (re)Set Font-Color to Gray
                sb = new StringBuffer(); // Reset it
                sb.append('\u001b');
                sb.append('[');
                sb.append(39); // Color
                sb.append('m');
                System.out.print(sb.toString()); // Set it.
```

```
                return (rc);
        }
}

/*
 * Name:    StdOut.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 15.05.2001 Tobias Kranz fixed "double newline"-bug
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: io package.
 */
package jftprefactoring.flexibilityandextensibility.io;

/**
 * Represents the standard output channel (StdOut)
 *
 * @since 0.0.2pA1
 */
/**
 * 1.9.1. Inline Class: JFTPIO and StdIO are in-lined into StdOut and
 * inheritance relationship is directly moved to StdOut. 1.9.2. Inheritance
 * relationship is removed due to its limited usage. 1.14. Move Field: isCli
 * local variable is copied from JFTPSuper to replace inheritance.
 */
public class StdOut {

        /**
         * 1.14. Instance variable replaces inheritance relationship.
         */
        boolean isCli;

        /**
         * 1.14. StdOut constructors.
         */
        public StdOut() {
                this.isCli = false;
        }

        public StdOut(boolean isCli) {
                this.isCli = isCli;
        }

        /**
         * Prints an object to StdOut.
         *
         * @param o
         *            Object to print
         *@since 0.0.2pA1
         */
        public void print(Object o) {
                if (this.isCli) {
                        System.out.print(o);
                } else {
                        // Insert GUI-Code here
```

```
		}
	}

	/**
	 * Prints a double to StdOut.
	 *
	 * @param d
	 *            double to print
	 *@since 0.0.2pA1
	 */
	public void print(double d) {
		if (this.isCli) {
			System.out.print(d);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a float to StdOut.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void print(float f) {
		if (this.isCli) {
			System.out.print(f);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a long to StdOut.
	 *
	 * @param l
	 *            long to print
	 *@since 0.0.2pA1
	 */
	public void print(long l) {
		if (this.isCli) {
			System.out.print(l);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints an int to StdOut.
	 *
	 * @param i
	 *            int to print
	 *@since 0.0.2pA1
	 */
	public void print(int i) {
		if (this.isCli) {
			System.out.print(i);
		} else {
			// Insert GUI-Code here
		}
	}
```

```
	/**
	 * Prints a short to StdOut.
	 *
	 * @param s
	 *            short to print
	 *@since 0.0.2pA1
	 */
	public void print(short s) {
		if (this.isCli) {
			System.out.print(s);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a byte to StdOut.
	 *
	 * @param b
	 *            byte to print
	 *@since 0.0.2pA1
	 */
	public void print(byte b) {
		if (this.isCli) {
			System.out.print(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a char to StdOut.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void print(char c) {
		if (this.isCli) {
			System.out.print(c);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a boolean to StdOut.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void print(boolean b) {
		if (this.isCli) {
			System.out.print(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints an object and "\n\r" to StdOut.
	 *
```

```
         * @param o
         *            Object to print
         *@since 0.0.2pA1
         *@version 0
         */
        public void println(Object o) {
                if (this.isCli) {
                        if  ((char)  o.toString().charAt(o.toString().length()  -
1) == '\n')
                                System.out.print(o);
                        else
                                System.out.println(o);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints a double and "\n\r" to StdOut.
         *
         * @param d
         *            double to print
         *@since 0.0.2pA1
         */
        public void println(double d) {
                if (this.isCli) {
                        System.out.println(d);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints a float and "\n\r" to StdOut.
         *
         * @param f
         *            float to print
         *@since 0.0.2pA1
         */
        public void println(float f) {
                if (this.isCli) {
                        System.out.println(f);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints a long and "\n\r" to StdOut.
         *
         * @param l
         *            long to print
         *@since 0.0.2pA1
         */
        public void println(long l) {
                if (this.isCli) {
                        System.out.println(l);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
```

```
	 * Prints an int and "\n\r" to StdOut.
	 *
	 * @param i
	 *            int to print
	 *@since 0.0.2pA1
	 */
	public void println(int i) {
		if (this.isCli) {
			System.out.println(i);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a short and "\n\r" to StdOut.
	 *
	 * @param s
	 *            short to print
	 *@since 0.0.2pA1
	 */
	public void println(short s) {
		if (this.isCli) {
			System.out.println(s);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a byte and "\n\r" to StdOut.
	 *
	 * @param b
	 *            byte to print
	 *@since 0.0.2pA1
	 */
	public void println(byte b) {
		if (this.isCli) {
			System.out.println(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a char and "\n\r" to StdOut.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void println(char c) {
		if (this.isCli) {
			System.out.println(c);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a boolean and "\n\r" to StdOut.
	 *
	 * @param b
```

```
         *            boolean to print
         *@since 0.0.2pA1
         */
        public void println(boolean b) {
               if (this.isCli) {
                      System.out.println(b);
               } else {
                      // Insert GUI-Code here
               }
        }
}

/*
 * Name:    CommandLineParser.java (extends Parser (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation.
 * 0.0.2pA2 03.05.2001 Martin Loh   GET implemented
 * 0.0.2pA4 15.05.2001 Tobias Kranz fixed 'containsWildcards(String, int)'
&&
 *                                  'parse(String)'
 * 0.0.2pA4 05.06.2001 Tobias Kranz added 'license' command
 * 0.1.99   10.06.2001 Tobias Kranz added possibility to set Tx modes.
 * 0.1.99t2 17.07.2001 Tobias Kranz 2nd fix of 'containsWildcards(String,
int)'
 *                                  added 'createPattern(String)' &&
 *                                            'comparePatternWithList(String,
String[])',
 *                                  implemented MGET and MPUT and the
 *                                  "local dir list" command-group.
 * 0.1.99t4 19.07.2001 Tobias Kranz added 'void pager(String)' && fixed bug
with
 *                                                    MGET && added 'String
getKnownCommands()' &&
 *                                       implemented MKDIR && LMKDIR && PWD &&
LPWD
 * 0.1.99t5 23.07.2001 Tobias Kranz improved lcd
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: parsers package.
 */
package jftprefactoring.flexibilityandextensibility.parsers;

import java.io.IOException;
import java.util.NoSuchElementException;
import java.util.StringTokenizer;
import java.util.Vector;

import jftprefactoring.flexibilityandextensibility.core.FTPASCIIData;
import jftprefactoring.flexibilityandextensibility.core.FTPAuthentication;
import jftprefactoring.flexibilityandextensibility.core.FTPBinaryData;
import jftprefactoring.flexibilityandextensibility.core.FTPConnection;
import jftprefactoring.flexibilityandextensibility.core.FTPDataType;
import jftprefactoring.flexibilityandextensibility.core.FTPDirectory;
import jftprefactoring.flexibilityandextensibility.core.FTPTransfer;
import jftprefactoring.flexibilityandextensibility.io.StdErr;
import jftprefactoring.flexibilityandextensibility.io.StdIn;
```

```
import jftprefactoring.flexibilityandextensibility.io.StdOut;

/**
 * Parses the CommandLine for known FTP-Commands.<BR>
 * This class is used by JFTP.<BR>
 *
 * @since 0.0.2pA1
 */
/**
 * 1.2.3.3. A set of local variables of JFTPSuper to cover jftpSuper.
 */
public class CommandLineParser extends Parser {

	private final char[] listLong1 = { 'l' };
	private final char[] listLong2 = { 'l', 's', ' ', '-', 'l' };
	private final char[] listShort1 = { 'l', 's' };
	private final char[] listShort2 = { 'd', 'i', 'r' };
	// the same for local
	private final char[] llistLong1 = { 'l', 'l' };
	private final char[] llistLong2 = { 'l', 'l', 's', ' ', '-', 'l' };
	private final char[] llistShort1 = { 'l', 'l', 's' };
	private final char[] llistShort2 = { 'l', 'd', 'i', 'r' };

	/**
	 *   1.2.3.3.   A   set   of   local   variables   of   FTPConnection,
FTPAuthentication,
	 * FTPTransfer, FTPDirectory and FTPDataType.
	 */
	private FTPConnection ftpConnection;
	private FTPAuthentication ftpAuthentication;
	private FTPTransfer ftpTransfer;
	private FTPDirectory ftpDirectory;
	private FTPDataType ftpASCIIData;
	private FTPDataType ftpBinaryData;

	/**
	 * Constructor
	 *
	 * @since The beginning of Time.
	 */
	public CommandLineParser() {
		this.ftpConnection = new FTPConnection();
		this.ftpAuthentication = new FTPAuthentication();
		this.ftpTransfer = new FTPTransfer();
		this.ftpDirectory = new FTPDirectory();
		this.ftpASCIIData = new FTPASCIIData();
		this.ftpBinaryData = new FTPBinaryData();

	}

	public CommandLineParser(FTPConnection ftpConnection) {
		this.ftpConnection = ftpConnection;
		this.ftpAuthentication = new FTPAuthentication(ftpConnection
				.getServer(), ftpConnection.getServerPort());
		this.ftpTransfer = new FTPTransfer(ftpConnection.getServer(),
				ftpConnection.getServerPort());
		this.ftpDirectory = new FTPDirectory(ftpConnection.getServer(),
				ftpConnection.getServerPort());
		this.ftpASCIIData = new FTPASCIIData(ftpConnection.getServer(),
				ftpConnection.getServerPort());
		this.ftpBinaryData                    =                    new
FTPBinaryData(ftpConnection.getServer(),
				ftpConnection.getServerPort());
```

```
	}

	/**
	 * 1.11. Remove Assignments to Parameters: assignments of parameter of
parse
	 * method are removed by using temporary variable.
	 */

	/**
	 * Parses the CommandLine.
	 *
	 * @param s
	 *            String to parse.
	 *@version 6
	 *@since 0.0.2pA1
	 */
	public void parse(String s) {
		try {

			/*
			 * Have you ever wondered how the parsing works? There
are 3 main
			 *  parts: At the first point we are checking for
commands that can
			 * be entered in any situation. At the second point we
are looking
			 *  for commands that only makes sense if we're
connected. And at
			 * least there's a section at the end where all command
are handled
			 * which may entered in an unconnected status.
			 */

			/**
			 * 1.11.1. Temporary variable is set value of parameter.
1.11.2. All
			 *  assignments to parameter are set to temporary
variable. 4.3.
			 * Rename Variable: temp variable is renamed to command.
			 */
			String command = s;
			command = command.trim();

			if (ftpConnection.getDebugLevel() >= 2)
				printErrorLine("parsing \"" + command + "\"");

			// OPENs a connection
			if (isOpenCommand(command)) {
				this.logon(command);
			} else if (isInfoCommand(command)) {
				ftpConnection.printLongJFTPInfo();
			} else if (isLicenseCommand(command)) {
				pager(ftpConnection.getLicense());
			} else if (isQuitExitCommand(command)) {
				printOutputLine("obsolete. Use 'bye' instead.");
			}
			// LocalLiSt short
			else if (isShortLocalListCommand(command)) {
				printOutput(ftpDirectory.getLocalShortDirList());
			}
			// LocalLiSt Long
			else if (isLongLocalListCommand(command)) {
				printOutput(ftpDirectory.getLocalLongDirList());
```

```
                    }
                    // HELP
                    else if (isHelpCommand(command)) {
                           String tmp;
                           tmp = getKnownCommands();

                           if (ftpConnection.isConnected()) {
                                  tmp += "Server Commands:\n";
                                  ftpTransfer.send(command);  //  Sends  'help'
to the server
                                  tmp += ftpTransfer.readCtrl(); // Reads the
servers
                                  // commands.
                           }
                           pager(tmp);
                    }
                    // LMKDIR
                    else if (isMakeLocalDirectoryCommand(command)) {
                           if
(ftpDirectory.createLocalDir(command.substring(7)))
                                  printOutputLine("Okay,   directory   \""   +
command.substring(7)
                                                + "\" was created.");
                           else
                                  printOutputLine("Unable    to    create    a
directory.");
                    }
                    // LRMDIR
                    else if (isRemoveLocalDirectoryCommand(command)) {
                           if
(ftpDirectory.removeLocalDir(command.substring(7)))
                                  printOutputLine("Okay,   directory   \""   +
command.substring(7)
                                                + "\" was removed.");
                           else
                                  printOutputLine("Unable   to   remove   that
directory.");
                    }
                    // LPWD
                    else if (isPrintLocalDirectoryCommand(command)) {

      printOutputLine(ftpDirectory.printLocalWorkingDir());
                    }
                    // LCD
                    else if (isChangeLocalDirectoryCommand(command)) {
                           if
(!ftpDirectory.changeLocalWorkingDir(command.substring(4)))
                                  printOutputLine("Can't  change  working  dir
to: "
                                                + command.substring(4));
                           else
                                  printOutputLine("Okay,      working      dir
changed.");
                    }

                    //  The  following  if's  are  only  interresting  if  we're
connected
                    else if (ftpConnection.isConnected()) {
                           // CLOSEs a connection
                           if (isCloseCommand(command)) {
                                  ftpConnection.disconnect();
                           }
                           // Change working Directory
```

```
                        else if (isChangeDirectoryCommand(command)) {

      ftpDirectory.changeRemoteWorkingDir(command.substring(3)
                                          .trim());
                        }
                        // LiSt short
                        else if (isShortListCommand(command)) {

      printOutput(ftpDirectory.getShortDirList());
                        }
                        // LiSt Long
                        else if (isLongListCommand(command)) {
                              printOutput(ftpDirectory.getLongDirList());
                        }
                        // GET
                        else if (isGetCommand(command)) {
                              if      (containsWildcards(command,      4,
command.length())) {
                                    printOutputLine("Wildcards   are   NOT
allowed. Use 'mget' instead.");
                              } else {
                                    if
(!ftpTransfer.getFile(command.substring(4).trim(),

      command.substring(4).trim()))
                                          printErrorLine("Unable  to  get
the file.");
                              }
                        }
                        // MGET
                        else if (isMGetCommand(command)) {
                              String[]               tmp               =
comparePatternWithList(createPattern(command
                                          .substring(4).trim()),
ftpDirectory
                                          .getDirListArray());
                              ftpTransfer.getFiles(tmp, tmp);
                        }
                        // PUT
                        else if (isPutCommand(command)) {
                              if      (containsWildcards(command,      4,
command.length())) {
                                    printOutputLine("Wildcards   are   NOT
allowed. Use 'mput' instead.");
                              } else {
                                    if
(!ftpTransfer.putFile(command.substring(4)))
                                          printOutputLine("Unable to put
the file.");
                              }
                        }
                        // MPUT
                        else if (isMPutCommand(command)) {
                              String[]               tmp               =
comparePatternWithList(createPattern(command
                                          .substring(4).trim()),
ftpDirectory
                                          .getLocalDirListArray());
                              ftpTransfer.putFiles(tmp);
                        }
                        // setting the TX mode
                        else if (isTypeCommand(command)) {
                              if (command.equals("bin"))
```

```
                                        ftpBinaryData.setTxMode(command);
                                  if (command.equals("asc"))
                                        ftpASCIIData.setTxMode(command);
                            }
                            // MKDIR
                            else if (isMakeDirectoryCommand(command)) {

      ftpDirectory.createRemoteDir(command.substring(6));
                            }
                            // RMDIR
                            else if (isRemoveDirectoryCommand(command)) {

      ftpDirectory.removeRemoteDir(command.substring(6));
                            }
                            // PWD
                            /*
                             * Yes, this costs performance, but its easier to
change the
                             *  progx  behavior,  for  example  when  adapting
another protocol.
                             */
                            else if (isPrintDirectoryCommand(command)) {

      printOutputLine(ftpDirectory.printRemoteWorkingDir());
                            } else if (isByeCommand(command)) {
                                  ftpConnection.disconnect();
                                  System.exit(0); // EXIT(0)
                            } else {
                                  if  (command.length()  >  0)  //  prevents
errors if you just
                                  // type
                                  // '\n'
                                  {
                                        if (ftpConnection.getDebugLevel() >=
2)
                                              printOutputLine("sending: " +
command);
                                        ftpTransfer.send(command);

      printOutputLine(ftpTransfer.readCtrl());
                                  }
                            }
                      } else // At this point we are not connected:
                      {
                            if (isByeCommand(command)) {
                                  System.exit(0); // EXIT(0)
                            } else {
                                  if  (command.length()  >  0)  //  prevents
errors if you just
                                  // type
                                  // '\n'
                                  {
                                        printErrorLine("Unable  to  execute  "
+ command + " .");
                                  }
                            }
                      }
                } catch (StringIndexOutOfBoundsException siobe) {
                      if (ftpConnection.getDebugLevel() >= 2)
                            printErrorLine("Error  in  main-loop  of  Clp:  " +
siobe.toString());
                } catch (NullPointerException npe) {
                      if (ftpConnection.getDebugLevel() >= 2)
```

```
                        printOutputLine("NullPointerException caugth (in
clp)");
            } catch (NoSuchElementException nsee) {
                  if (ftpConnection.getDebugLevel() >= 2)
                        printOutputLine("NoSuchElementException caugth (in
clp)");
            }
      }

      /**
       * Will be implemented when j2sdk1.4 is released!
       */
      /*
       * IS implemented! But in FTPCmdServer.\ private void getFiles(String
       * pattern) { }
       */

      /**
       * Manages the whole authentication/logon procedure
       *
       * @param s
       *            String containing all params entered
       *@since 0.0.2pA4
       *@version 2
       */
      private void logon(String s) {
            boolean serverIsValid = false;
            String server;

            /*
             * s.substring(5) will 'return false' if s.length < 5 !!!
             */
            if (s.length() > 5)
                  server = s.substring(5).trim();
            else
                  server = "";

            if (ftpConnection.getDebugLevel() >= 2)
                  printErrorLine("server=\"" + server + "\"");

            if (ftpConnection.isConnected()) {
                  printOutputLine("Unable to re-connect; You must
disconnect first...");
            } else {
                  while (!serverIsValid) {
                        if (server.length() > 4)// What's the min length
of a valid inet
                        // Address?
                        {
                              serverIsValid = true;
                        } else {
                              printOutput("(host): ");
                              try {
                                    server = readInputLine();
                              } catch (IOException ioe) {
                              }
                        }
                  }

                  if (ftpConnection.getDebugLevel() >= 2)
                        printOutputLine("contacting server " + server);

                  if (ftpConnection.setServerTo(server)) // setting (new)
```

```
Server
                    // (Server
                    // is _really_ there...:~) )
                    {
                           ftpConnection.connect(); // CONNECT !
                           if (ftpConnection.isConnected()) // connected ?
                           {
                                  // if (ftpConnection.getDebugLevel() >= 2)
                                  printOutputLine(ftpTransfer.readCtrl());

                                  if (!ftpAuthentication.authenticate()) {
                                         /*
                                          *     Should     be     handled     by
authenticate.
                                          */
                                         //     printErrorLine("Unable     to
authenticate.");
                                         parse("close"); // close connection
from Server
                                  }
                           }
                    } else {
                           printErrorLine("Cannot connect to server.");
                    }
             }
       }

       /**
        *   3.1.   Add   Parameter:   integer   end   parameter   is   added   to
containsWildcards
        * method.
        */

       /**
        * Checks if the given String contains any wildcards ('*'/'?').
        *
        * @param s
        *            String to check
        *@param start
        *            int position to start at.
        *@return true if the given String contains any wildcards; otherwise
not
        *@since 0.0.2pA3
        */
       private boolean containsWildcards(String s, int start, int end) {
             boolean rc = false;
             int i;

             // Check for a valid range
             if (start <= s.length()) {
                    for (i = start; i < end; i++) {
                           if   (((char)   s.charAt(i)   ==   '*')   ||   ((char)
s.charAt(i) == '?')
                                         || ((char) s.charAt(i) == '[')) {
                                  rc = true;
                                  break;
                           }
                    }
             }

             return (rc);
       }
```

```
	/**
	 * Creates a regular-expression-pattern usable by java.lang.util.regex
from
	 * any posix-regex. Example: Posix-regex: java.lang.util.regex: (see
	 * j2sdk1.4-doc) *[0-9]t.ab? {graph}*[0-9]t.ab{graph}? ||
	 * {graph}*{num}?t.ab{graph}?
	 *
	 * @param pattern
	 *            The posix-regex to compute
	 *@return The matching java.lang.regex-regex
	 *@since 0.1.99t2
	 *@version 1
	 */
	private String createPattern(String pattern) {
		// Initial things
		StringTokenizer st;
		String pat = pattern, // local-temporary pattern
		rc = "";

		if (ftpConnection.getDebugLevel() >= 2)
			printErrorLine("Pattern=" + pattern);
		/* <Parsing for '''s> */
		st = new StringTokenizer(pat, "*");

		if (st.hasMoreTokens())
			rc += st.nextToken();

		while (st.hasMoreTokens())
			rc += "{graph}*" + st.nextToken();

		// Check for start and end
		if (pat.charAt(0) == '*')
			rc = "{graph}*" + rc;

		if ((pattern.charAt(pattern.length() - 1) == '*'))
			rc += "{graph}*";
		/* </Parsing for '''s> */

		// Resetting
		st = null;
		pat = rc;
		rc = "";

		/* <Parsing for '?''s> */
		st = new StringTokenizer(pat, "?");

		if (st.hasMoreTokens())
			rc += st.nextToken();

		while (st.hasMoreTokens())
			rc += "{graph}?" + st.nextToken();

		// Check for start and end
		if (pat.charAt(0) == '?')
			rc = "{graph}?" + rc;

		if ((pattern.charAt(pattern.length() - 1) == '?'))
			rc += "{graph}?";
		/* </Parsing for '?''s> */

		/*
		 * Should be used before running another loop on the pattern.
//
```

```
                * Resetting st = null; pat = rc; rc = "";
                */
               return (rc);
        }

        /**
         * Compares a java.lang.String array with a given java.util.regex
compatible
         * regular expression and returns a list of all matching entries.
         *
         * @param pattern
         *            The java.util.regex pattern to use.
         *@param list
         *            A java.lang.String array which should be compared to the
         *            pattern
         *@return A java.lang.String array which all entries from list that
are
         *         matching the pattern.
         *@since 0.1.99t2
         *@version 1
         */
        private String[] comparePatternWithList(String pattern, String[] list)
{
               int i;
               java.util.regex.Pattern                    p                    =
java.util.regex.Pattern.compile(pattern);
               java.util.regex.Matcher m;
               Vector<String> tmp = new Vector<String>(1);

               for (i = 0; i < list.length; i++) {
                      m = p.matcher(list[i]);

                      if (m.matches())
                             tmp.add((String) list[i]);
               }

               // Vector tmp -> String rc[n]
               String[] rc = new String[tmp.size()];

               for (i = 0; i < tmp.size(); i++)
                      rc[i] = tmp.elementAt(i);

               return (rc);
        }

        /**
         * 1.11. Remove Assignments to Parameters: assignments of parameter of
pager
         * method are removed by using temporary variable.
         */

        /**
         * Show any txpe of text side by side. A pager. This method looks for
'\n''s
         * (24 times), then shows a "Press return.."-msg and waits for any
         * return-terminated input and restarts that loop until EOF.
         *
         * @param text
         *            The text to display
         *@since 0.1.99.test4
         *@version 3
         */
        private void pager(String text) {
```

```
              int i, j = 0;
              char c;

              /**
               *  1.11.1.  Temporary  variable  is  set  value  of  parameter.
1.11.2. All
               * assignments to parameter are set to temporary variable.
               */
              String temp = text;
              temp += "\n"; // Ugly way to make sure the prompt comes up in a
new
              // line.

              while (temp.length() > j) {
                     for (i = 0; i < 23; i++) {
                            do {
                                   c = temp.charAt(j++);
                                   printOutput(c + "");
                            } while (c != '\n');
                     }

                     printOutput("\nPress [Return] to continue...");
                     try {
                            readInputLine();
                     } catch (IOException whoCares) {
                     }
              }
       }

       /**
        * Returns a java.lang.String containing all commands known by 'this'.
        *
        * @return A java.lang.String containing all commands.
        *@since 0.1.99test4
        *@version 1
        */
       private String getKnownCommands() {
              return    ("Commands    recognized    by    JavaFTP:    (*    =>
unimplemented)\n"
                           + "\n"
                           + "Conntection-commands:\n"
                           + "\n"
                           + "open           Asks you the server's name or ip
to connect to.\n"
                           + "open SERVER    Opens a connection to the server
SERVER.\n"
                           + "close          Closes the active connection.\n"
                           + "\n"
                           + "List-commands:\n"
                           + "l\n"
                           + "ls -la            Shows a detailed list of the
active directory "
                           + "on the server.\n"
                           + "\n"
                           + "ls\n"
                           + "dir            Shows a short list of the active
directory on the server.\n"
                           + "\n"
                           + "ll\n"
                           + "lls -la           Shows a detailed list of the
active directory on "
                           + "the localhost.\n"
                           + "\n"
```

```
                            + "lls\n"
                            + "ldir           Shows a short list of the active
directory on "
                            + "the localhost.\n"
                            + "\n"
                            + "Options and switches:\n"
                            +  "type  X               Sets  the  Tx-mode  to  X.
(X=i(binary)||X=a(ascii))\n"
                            + "asc            Sets the Tx-mode to ascii.\n"
                            + "bin            Sets the Tx-mode to binary.\n"
                            + "\n"
                            + "Directory relative commands:\n"
                            + "cd PATH         Changes the active directory on
the server to PATH.\n"
                            + "lcd PATH        Changes the active directory on
the localhost to PATH.\n"
                            +  "pwd                 Shows  the  current  working
directory on the server.\n"
                            +  "lpwd                Shows  the  current  working
directory on the localhost.\n"
                            + "mkdir DIR      Creates a directory named DIR on
the server.\n"
                            + "rmdir DIR      Removes a directory named DIR on
the server.\n"
                            + "lmkdir DIR     Creates a directory named DIR on
the localhost.\n"
                            + "lrmdir DIR     Removes a directory named DIR on
the localhost.\n"
                            + "\n"
                            + "Transfer-commands:\n"
                            + "get FILE       Gets a file named FILE\n"
                            + "put FILE       Puts a file named FILE\n"
                            + "mget PATTERN   Gets all files that matches the
pattern PATTERN.\n"
                            + "mput PATTERN   Puts all files that matches the
pattern PATTERN.\n"
                            + "\n"
                            + "Misc stuff:\n"
                            +  "info                Shows  a  short  info  about
JavaFTP.\n"
                            + "license        Shows the license under "
                            + "wich you are allowed to use JavaFTP.\n"
                            + "help           Shows this helpscreen.\n"
                            + "bye              Quits the program and closses
all open connections.\n"
                            + "\n");
      }

      /**
       *
       *  1.5.  Hide  Delegate:  StdIn,  StdOut  and  StdErr  are  hidden  by
readInputLine,
       * printOutput, printOutputLine and printErrorLine methods.
       */
      private String readInputLine() throws IOException {
            StdIn stdin = new StdIn();
            return stdin.readLine();
      }

      private void printOutput(String message) {
            StdOut stdout = new StdOut();
            stdout.print(message);
      }
```

```
	private void printOutputLine(String message) {
		StdOut stdout = new StdOut();
		stdout.println(message);
	}

	private void printErrorLine(String message) {
		StdErr stderr = new StdErr();
		stderr.println(message);
	}

	/**
	 * 3.4. Decompose Conditional: isOpenCommand method.
	 */
	private boolean isOpenCommand(String command) {
		return (command.startsWith("open"));
	}

	/**
	 * 3.4. Decompose Conditional: isInfoCommand method.
	 */
	private boolean isInfoCommand(String command) {
		return (command.equals("info"));
	}

	/**
	 * 3.4. Decompose Conditional: isLicenseCommand method.
	 */
	private boolean isLicenseCommand(String command) {
		return (command.equals("license"));
	}

	/**
	 *  3.4.  Decompose  Conditional:  isQuitCommand  method.  4.2.  Rename
Method:
	 * isQuitCommand is renamed to isQuitExitCommand.
	 */
	private boolean isQuitExitCommand(String command) {
		return ((command.equals("quit")) || (command.equals("exit")));
	}

	/**
	 * 3.4. Decompose Conditional: isHelpCommand method.
	 */
	private boolean isHelpCommand(String command) {
		return (command.equals("help"));
	}

	/**
	 * 3.4. Decompose Conditional: isShortLocalListCommand method.
	 */
	private boolean isShortLocalListCommand(String command) {
		return (command.equals(String.valueOf(llistShort1)) || command
				.equals(String.valueOf(llistShort2)));
	}

	/**
	 * 3.4. Decompose Conditional: isLongLocalListCommand method.
	 */
	private boolean isLongLocalListCommand(String command) {
		return (command.equals(String.valueOf(llistLong1)) || command
				.equals(String.valueOf(llistLong2)));
	}
```

```
	/**
	 * 3.4. Decompose Conditional: isMakeLocalDirectoryCommand method.
	 */
	private boolean isMakeLocalDirectoryCommand(String command) {
		return (command.startsWith("lmkdir"));
	}

	/**
	 * 3.4. Decompose Conditional: isRemoveLocalDirectoryCommand method.
	 */
	private boolean isRemoveLocalDirectoryCommand(String command) {
		return (command.startsWith("lrmdir"));
	}

	/**
	 * 3.4. Decompose Conditional: isPrintLocalDirectoryCommand method.
	 */
	private boolean isPrintLocalDirectoryCommand(String command) {
		return (command.equals("lpwd"));
	}

	/**
	 * 3.4. Decompose Conditional: isChangeLocalDirectoryCommand method.
	 */
	private boolean isChangeLocalDirectoryCommand(String command) {
		return (command.startsWith("lcd"));
	}

	/**
	 * 3.4. Decompose Conditional: isCloseCommand method.
	 */
	private boolean isCloseCommand(String command) {
		return (command.equals("close"));
	}

	/**
	 * 3.4. Decompose Conditional: isChangeDirectoryCommand method.
	 */
	private boolean isChangeDirectoryCommand(String command) {
		return (command.startsWith("cd"));
	}

	/**
	 * 3.4. Decompose Conditional: isShortListCommand method.
	 */
	private boolean isShortListCommand(String command) {
		return (command.equals(String.valueOf(listShort1)) || command
				.equals(String.valueOf(listShort2)));
	}

	/**
	 * 3.4. Decompose Conditional: isLongListCommand method.
	 */
	private boolean isLongListCommand(String command) {
		return (command.equals(String.valueOf(listLong1)) || command
				.equals(String.valueOf(listLong2)));
	}

	/**
	 * 3.4. Decompose Conditional: isGetCommand method.
	 */
	private boolean isGetCommand(String command) {
```

```
                return (command.startsWith("get"));
        }

        /**
         * 3.4. Decompose Conditional: isMGetCommand method.
         */
        private boolean isMGetCommand(String command) {
                return (command.startsWith("mget"));
        }

        /**
         * 3.4. Decompose Conditional: isPutCommand method.
         */
        private boolean isPutCommand(String command) {
                return (command.startsWith("put"));
        }

        /**
         * 3.4. Decompose Conditional: isMPutCommand method.
         */
        private boolean isMPutCommand(String command) {
                return (command.startsWith("mput"));
        }

        /**
         * 3.4. Decompose Conditional: isTypeCommand method.
         */
        private boolean isTypeCommand(String command) {
                return  (command.startsWith("type")  ||  command.equals("bin")  ||
command
                                .equals("asc"));
        }

        /**
         * 3.4. Decompose Conditional: isMakeDirectoryCommand method.
         */
        private boolean isMakeDirectoryCommand(String command) {
                return (command.startsWith("mkdir"));
        }

        /**
         * 3.4. Decompose Conditional: isRemoveDirectoryCommand method.
         */
        private boolean isRemoveDirectoryCommand(String command) {
                return (command.startsWith("rmdir"));
        }

        /**
         * 3.4. Decompose Conditional: isPrintDirectoryCommand method.
         */
        private boolean isPrintDirectoryCommand(String command) {
                return (command.equals("pwd"));
        }

        /**
         * 3.4. Decompose Conditional: isByeCommand method.
         */
        private boolean isByeCommand(String command) {
                return (command.equals("bye"));
        }
}

/*
```

```
 * Name:    Parser.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: parsers package.
 */
package jftprefactoring.flexibilityandextensibility.parsers;

/**
 * Abstract class. Defines methods to parse Strings.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.2.3.2. Remove inheritance relationship to JFTPSuper which only rely on
 * jftpSuper and add a set of local variables of JFTPSuper.
 */
public abstract class Parser {
	public Parser() {
		boolean busy = false;
		System.out.println(busy);
		/*
		 * There was an error msg at compile-time that let me do such
strange
		 * things. ;-)
		 */
	}

	public abstract void parse(String s);
}

/*
 * Name:    ServerResponseParser.java (extends Parser (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: parsers package.
 */
package jftprefactoring.flexibilityandextensibility.parsers;

/**
 * Parses the FTPServers responses.
 *
 * @since 0.0.2pA1
 */
public class ServerResponseParser extends Parser {
	/**
```

```
         * Parses a given String for known FTPServer-replies.
         *
         * @param s
         *            String to parse
         *@since 0.0.2pA1
         */
        public void parse(String s) {
                /*
                 *  if  (s.equals(xxxxxx))  {  Insert  known  FTPServer  replies
here. }
                 */
        }
}

/*
 * Name:    Progressbar.java  (extens JFTPSuper.java)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99   11.06.2001 Tobias Kranz Creation
 * 0.1.99t5 23.07.2001 Tobias Kranz Added Comments
 */
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 6.1. Extract Package: ui package.
 */
package jftprefactoring.flexibilityandextensibility.ui;

/**
 * Shows a textual progressbar
 *
 * @since 0.1.99
 *@version 1
 */
/**
 * 1.2.3.2. Remove inheritance relationship to JFTPSuper which only rely on
 * jftpSuper and add a set of local variables of JFTPSuper.
 */
public class Progressbar {
        private long max;
        private long pos = 1;

        /**
         * Constructor. Here you must set the maximum value the progressbar
may
         * have.
         *
         * @param max
         *            The maximum value the progressbar may have.
         *@since 0.1.99
         *@version 1
         */
        public Progressbar(int max) {
                this.max = max;
                this.show();
        }

        /**
         * Sets the progressbar to the given value.
```

```
         *
         * @param pos
         *            The current position of the progressbar
         *@since 0.1.99
         *@version 1
         */
        public void set(int pos) {
                if (pos <= max) {
                        this.pos = pos;
                }
                show();
        }

        /**
         * Shows the progressbar in it's current status
         *
         * @since 0.1.99
         *@version 1
         */
        private void show() {
                int i, stat = 0;
                String graphStat = "";
                double stat1 = ((double) 100 / max) * pos;

                String statstr = Double.toString(stat1);
                stat            =           Integer.parseInt(statstr.substring(0,
statstr.indexOf('.')));

                for (i = 0; i < (int) 20 * (((float) stat) / 100); i++)
                        graphStat += "="; // setting '=''s for reached %

                for (; i < 20; i++)
                        graphStat += " "; // filling the rest with ' ''s

                if ((int) ((float) stat) / 100 >= 1)
                        System.out.print("Progress: [" + graphStat + "]: " +
stat
                                        + "% complete (" + (int) pos + " bytes
transfered)\n");
                // else if ((int)((float)stat)/10 >= 1)
                //  System.out.print("Progress: ["+graphStat+"]:    "  +stat+"%
complete ("+
                // (int)pos/1024+" kb)\r");
                else
                        System.out.print("Progress: [" + graphStat + "]:    " +
stat
                                        + "% complete (" + (int) pos / 1024 + "
kb)\r");
        }
}

/*
 * Name:    JFTP.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name                      changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz              Creation
 * 0.0.2pA2 02.05.2001 Tobias Kranz              Added 'nextParamIsNumeric()'
 *                                               & 'nextParamExists()'
 * 0.0.2pA2 04.05.2001 Sebastian Schipper        Two bugs fixed
 */
/**
```

```
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
package jftprefactoring.flexibilityandextensibility;

import java.io.IOException;
import java.util.BitSet;

import jftprefactoring.flexibilityandextensibility.core.FTPAuthentication;
import jftprefactoring.flexibilityandextensibility.io.StdErr;
import jftprefactoring.flexibilityandextensibility.io.StdIn;
import jftprefactoring.flexibilityandextensibility.io.StdOut;
import
jftprefactoring.flexibilityandextensibility.parsers.CommandLineParser;
import jftprefactoring.flexibilityandextensibility.parsers.Parser;

/**
 * Main class.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.2.3.2. Remove inheritance relationship to JFTPSuper which only rely on
 *  jftpSuper  and  add  a  local  variable  of  JFTPSuper.  1.2.3.3.  A  local
variable of
 * JFTPSuper to cover jftpSuper.
 */
public class JFTP {

      private final char[] guiShort = { '-', 'g' };
      private final char[] cliShort = { '-', 'c' };
      private final char[] helpShort = { '-', 'h' };
      private final char[] portShort = { '-', 'P' };
      private final char[] passiveShort = { '-', 'p' };
      private final char[] verboseShort = { '-', 'v' };
      private final char[] anonymousShort = { '-', 'a' };
      private final char[] aPasswdShort = { '-', 'A', 'p' };
      private final char[] guiLong = { '-', '-', 'g', 'u', 'i' };
      private final char[] cliLong = { '-', '-', 'c', 'l', 'i' };
      private final char[] helpLong = { '-', '-', 'h', 'e', 'l', 'p' };
      private final char[] portLong = { '-', '-', 'P', 'o', 'r', 't' };
      private final char[] verboseLong = { '-', '-', 'v', 'e', 'r', 'b',
'o',
                  's', 'e' };
      private final char[] passiveLong = { '-', '-', 'p', 'a', 's', 's',
'i',
                  'v', 'e' };
      private final char[] anonymousLong = { '-', '-', 'a', 'n', 'o', 'n',
'y',
                  'm', 'o', 'u', 's' };
      private final char[] aPasswdLong = { '-', '-', 'A', 'n', 'o', 'n',
'y',
                  'm', 'o', 'u', 's', 'P', 'a', 's', 's', 'w', 'd' };

      /**
       * 1.14. Move Field: options local variable is moved from JFTPSuper to
       * replace inheritance.
       */
      private BitSet options = new BitSet(4);

      /**
       * 1.2.3.3. A local variable of FTPAuthentication.
       */
```

```
	private FTPAuthentication ftpAuthentication;

	private StdIn stdin;
	private StdOut stdout;
	private StdErr stderr;

	/**
	 * Sets initial values.
	 *
	 * @since 0.0.2pA1
	 */
	public JFTP() {
		this.ftpAuthentication = new FTPAuthentication();
		this.stdin = new StdIn();
		this.stdout = new StdOut();
		this.stderr = new StdErr();
		/*
		 * BitSet 'options'
		 *
		 * #: if true: default: 0 => cli true, 1 => passive true, 2 =>
anonymous
		 * false, 3 => autoconnect false.
		 */
		this.options.set(0);
		this.options.set(1);
		this.options.clear(2);
		this.options.clear(3);
	}

	/**
	 * Is Cli-mode running ?
	 *
	 * @return true if Cli-mode is running; false if not
	 *@since 0.0.2pA1
	 */
	protected boolean isCli() {
		return ((boolean) options.get(0));
	}

	/**
	 * Sets Cli-mode to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setCli() {
		options.set(0);
	}

	/**
	 * Sets Cli-mode to false.
	 *
	 * @since 0.0.2pA1
	 */
	protected void unsetCli() {
		options.clear(0);
	}

	/**
	 * Is Gui-mode running ?
	 *
	 * @return true if isCli() returns false and other way round.
	 *@since 0.0.2pA1
	 */
```

```
	protected boolean isGui() {
		return (((boolean) options.get(0) ? (false) : (true)));
	}

	/**
	 * Same than unsetCli.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setGui() {
		unsetCli();
	}

	/**
	 * Is passive connection mode prefered ?
	 *
	 * @return true if passive mode is prefered; false if not
	 *@since 0.0.2pA1
	 */
	protected boolean isPassive() {
		return ((boolean) options.get(1));
	}

	/**
	 * Sets passive mode to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setPassive() {
		options.set(1);
	}

	/**
	 * Sets passive mode to false.
	 *
	 * @since 0.0.2pA1
	 */
	protected void unsetPassive() {
		options.clear(1);
	}

	/**
	 * Is anonymous logon prefered ?
	 *
	 * @return true if anonymous logon is prefered; otherwise false
	 *@since 0.0.2pA1
	 */
	protected boolean isAnonymous() {
		return ((boolean) options.get(2));
	}

	/**
	 * Sets anonymous logon to true.
	 *
	 * @since 0.0.2pA1
	 */
	protected void setAnonymous() {
		options.set(2);
	}

	/**
	 * Sets anonymous logon to false.
	 *
```

```
         * @since 0.0.2pA1
         */
        protected void unsetAnonymous() {
                options.clear(2);
        }

        /**
         * Returns Autoconnect
         *
         * @return true if Autoconnect is true; otherwise false
         *@since 0.0.2pA2
         */
        protected boolean isAutoconnect() {
                return ((boolean) options.get(3));
        }

        /**
         * Sets Autoconnect to true.
         *
         * @since 0.0.2pA2
         */
        protected void setAutoconnect() {
                options.set(3);
        }

        /**
         * Sets Autoconnect to false.
         *
         * @since 0.0.2pA2
         */
        protected void unsetAutoconnect() {
                options.clear(3);
        }

        /**
         * Starts the prog.
         *
         * @param CommandLineArguments
         *                .
         *@since 0.0.2pA1
         */
        public static void main(String[] args) {
                JFTP j = new JFTP();
                j.parseArgs(args); // scanning cmdLine

                if (j.isCli())
                        j.cli();
                else
                        j.gui();
        }

        /**
         * Starts the CommandLineInterface (cli) version.
         *
         * @since 0.0.2pA1
         */
        private void cli() {
                Parser clp = new CommandLineParser();

                if (ftpAuthentication.getDebugLevel() >= 1)
                        printOutputLine("DebugLevel:              "              +
ftpAuthentication.getDebugLevel());
                printOutputLine(ftpAuthentication.getJFTPInfo());
```

```
                printOutputLine(ftpAuthentication.getCopyright());
                printOutputLine(ftpAuthentication.getLicenseInfo());

                if (isAutoconnect()) {
                        clp.parse("open " + ftpAuthentication.getServerIP());
                }

                while (true) {
                        printOutput("jftp> ");

                        try {
                                clp.parse(readInputLine());
                        } catch (IOException e) {
                                printErrorLine("Couldn't read from StdIn!");
                        }
                }
        }

        /**
         * Starts the GUI version.
         *
         * @since 0.0.2pA1
         */
        private void gui() {
                // GUI gui1 = new GUI();
        }

        /**
         * Parses the commandLine arguments.
         *
         * @param a
         *            String array to parse
         *@since 0.0.2pA1
         *@version 1
         */
        private void parseArgs(String[] a) {
                int i;

                try {
                        for (i = 0; i < a.length; i++) {
                                if (isHelpCommand(a[i])) {
                                        this.showHelp();
                                } // Help
                                else if (isPassiveCommand(a[i])) {
                                        setPassive();
                                } // Passive mode
                                else if (isAnonymousCommand(a[i])) {
                                        setAnonymous();
                                } // Anonymous mode
                                else  if  (isAnonymousPasswordCommand(a[i]))  {  //
Anonymous
                                        // Passwd
                                        if (isAnonymous()) {
                                                if (this.nextParamExists(a, i))

        ftpAuthentication.setAnonymousPasswdTo(a[++i]);
                                                else
                                                        this.showHelp();
                                        } else {
                                                printErrorLine("You   have   set   the
anonymous password but not anonymous login!?!?");
                                                printErrorLine("Exiting   with   error
(1).");
```

```
                                        System.exit(1);
                                }

                                if (ftpAuthentication.getDebugLevel() >= 2)
                                        printOutputLine("Anonymous  Password:
"
                                                        +
ftpAuthentication.getAnonymousPasswd());
                        } else if (isGUICommand(a[i])) {
                                setGui();
                        } // GUI
                        else if (isCLICommand(a[i])) {
                                setCli();
                        } // CLI
                        else if (isVerbosityCommand(a[i])) { // Verbosity
                                if      ((i      <      a.length)      &&
(this.nextParamIsNumeric(a, i))) {
                                        short     tmpDebug     =     (short)
Integer.parseInt(a[++i]);
                                        if ((tmpDebug <= 2) && (tmpDebug >=
0))

      ftpAuthentication.setDebugLevel(tmpDebug);
                                } else {
                                        printErrorLine("Missing   or   wrong
parameter.");
                                        this.showHelp();
                                }
                        } // Port
                        else if (isPortCommand(a[i])) {
                                if      ((i      <      a.length)      &&
(this.nextParamIsNumeric(a, i))) {
                                        int             tmpPort             =
Integer.parseInt(a[++i]);
                                        if ((tmpPort >= 1) && (tmpPort <=
65535)) // Ok !
                                        {

      ftpAuthentication.setServerPortTo(tmpPort);
                                                printOutputLine("Don't use std
port. Using "
                                                                +
ftpAuthentication.getServerPort()
                                                                + " instead.");
                                        } else {
                                                printOutputLine("!WARNING!
Specified port not in proper range.");
                                                printOutputLine("!WARNING!
Falling back to default port (21).");
                                        }
                                } else {
                                        printErrorLine("Missing   or   wrong
parameters");
                                        this.showHelp();
                                }
                        } else {
                                if (i == (a.length - 1)) // Last parameter
                                {
                                        if
(ftpAuthentication.setServerTo(a[i]))
                                                setAutoconnect();
                                } else {
                                        printOutputLine("Unknown  Option:  '"
```

```
+ a[i] + "'");
                                        this.showHelp();
                                  }
                            }
                      }

              } catch (Exception e) {
                    printOutput("Exception: ");
                    e.printStackTrace();
              }

              // if (ftpConnection.isPassive())
              // printOutputLine("PASSIVE mode prefered.");
              if (isAnonymous())
                    printOutputLine("We'll logon as \"anonymous\".");
              // if (ftpConnection.isCli()) printOutputLine("starting CLI.");
              if (isGui())
                    printOutputLine("starting GUI.");
       }

       /**
        * 1.11. Remove Assignments to Parameters: assignments of parameter of
        * nextParamIsNumeric method are removed by using temporary variable.
        */

       /**
        * Checks if the next parameter is a numeric value.
        *
        * @param a
        *            String array of parameters to check
        *@param pos
        *            current position in array a
        *@return true if the next parameter exists and is numeric;otherwise
false
        *@since 0.0.2pA2
        */
       private boolean nextParamIsNumeric(String[] a, int pos) {
              boolean rc = false;
              int tmp;

              /**
               *  1.11.1.  Temporary  variable  is  set  value  of  parameter.
1.11.2. All
               * assignments to parameter are set to temporary variable.
               */
              int temp = pos;
              try {
                    tmp = Integer.parseInt(a[(++temp)]);
                    writeLineToConsole(tmp + "");
                    rc = true;
              } catch (IndexOutOfBoundsException iobe) {
                    if (ftpAuthentication.getDebugLevel() >= 2)
                          printErrorLine("No more parameters.");
              } catch (NumberFormatException nfe) {
                    if (ftpAuthentication.getDebugLevel() >= 2)
                          printErrorLine("No more parameters.");
              }

              return rc;
       }

       /**
        * 1.11. Remove Assignments to Parameters: assignments of parameter of
```

```
         * nextParamExists method are removed by using temporary variable.
         */

        /**
         * Checks if the next parameter exists.
         *
         * @param a
         *            String array to check
         *@param pos
         *            current position in array a
         *@return true if the next parameter exists; else false
         *@since 0.0.2pA2
         */
        private boolean nextParamExists(String[] a, int pos) {
                boolean rc = true;

                /**
                 *  1.11.1.  Temporary  variable  is  set  value  of  parameter.
1.11.2. All
                 * assignments to parameter are set to temporary variable.
                 */
                int temp = pos;
                try {
                        String tmp = a[++temp];
                        writeLineToConsole(tmp);
                } catch (IndexOutOfBoundsException iobe) {
                        rc = false;
                }

                // maybe "(pos < a.size) ? (return true) : (return false)

                return rc;
        }

        /**
         * Shows the help screen.
         *
         * @since 0.0.1
         */
        private void showHelp() {
                /*
                 * Verbosity  level  >  1  should  only  be  used  for  development.
Stable
                 * versions should only contain an option for setting verbosity
to true
                 * or false.
                 */
                printOutputLine(ftpAuthentication.getJFTPInfo());
                printOutputLine("usage: JFTP [-h[c|g]pa] [Server]");
                printOutputLine("       JFTP [--help][--verbose X][--passive][-
-anonymous[--anonymousPasswd X]]");
                printOutputLine("            [--port X][Server]\n");
                printOutputLine("       -h,    --help                Shows this
help");
                printOutputLine("       -c,    --cli                 Starts the
CLI-version (Default).");
                printOutputLine("       -g,    --gui                 Starts the
GUI-version.");
                printOutputLine("         -p,     --passive               Force
passive mode ftp.");
                printOutputLine("                                       (Default is
active mode ftp(NOT now))");
                printOutputLine("       -a,    --anonymous          Causes jftp
```

```
to bypass normal login procedure");
            printOutputLine("                                          and use
anonymous login instead.");
            printOutputLine("       -Ap X, --AnonymousPasswd X  Causes jftp
to use X as anonymous password.");
            printOutputLine("        -v X,   --verbose X           Sets the
vorbosity level where X = 0 - 2.");
            printOutputLine("        -P X,   --Port X              Sets the
port number to X.");
            System.exit(0); // <- !EXIT!
      }

      /**
       * 1.6. Extract Method: readInputLine, printOutput, printOutputLine,
       * printErrorLine and writeLineToConsole methods.
       */
      private String readInputLine() throws IOException {
            return stdin.readLine();
      }

      private void printOutput(String message) {
            stdout.print(message);
      }

      private void printOutputLine(String message) {
            stdout.println(message);
      }

      private void printErrorLine(String message) {
            stderr.println(message);
      }

      protected void writeLineToConsole(String message) {
            System.out.println(message);
      }

      /**
       * 3.4. Decompose Conditional: isHelpCommand method.
       */
      private boolean isHelpCommand(String command) {
            return (command.equals(String.valueOf(helpShort)) || command
                        .equals(String.valueOf(helpLong)));
      }

      /**
       * 3.4. Decompose Conditional: isPassiveCommand method.
       */
      private boolean isPassiveCommand(String command) {
            return (command.equals(String.valueOf(passiveShort)) || command
                        .equals(String.valueOf(passiveLong)));
      }

      /**
       * 3.4. Decompose Conditional: isAnonymousCommand method.
       */
      private boolean isAnonymousCommand(String command) {
            return    (command.equals(String.valueOf(anonymousShort))    ||
command
                        .equals(String.valueOf(anonymousLong)));
      }

      /**
       * 3.4. Decompose Conditional: isAnonymousPasswordCommand method.
```

```
         */
        private boolean isAnonymousPasswordCommand(String command) {
                return (command.equals(String.valueOf(aPasswdShort)) || command
                                .equals(String.valueOf(aPasswdLong)));
        }

        /**
         * 3.4. Decompose Conditional: isGUICommand method.
         */
        private boolean isGUICommand(String command) {
                return (command.equals(String.valueOf(guiShort)) || command
                                .equals(String.valueOf(guiLong)));
        }

        /**
         * 3.4. Decompose Conditional: isCLICommand method.
         */
        private boolean isCLICommand(String command) {
                return (command.equals(String.valueOf(cliShort)) || command
                                .equals(String.valueOf(cliLong)));
        }

        /**
         * 3.4. Decompose Conditional: isVerbosityCommand method.
         */
        private boolean isVerbosityCommand(String command) {
                return (command.equals(String.valueOf(verboseShort)) || command
                                .equals(String.valueOf(verboseLong)));
        }

        /**
         * 3.4. Decompose Conditional: isPortCommand method.
         */
        private boolean isPortCommand(String command) {
                return (command.equals(String.valueOf(portShort)) || command
                                .equals(String.valueOf(portLong)));
        }
}
```

Source Code B-6: JFTP Refactored Source Code Using QARtF

```
/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.net.InetAddress;

/**
 * 1.3.1. Replace Conditional with Polymorphism: FTPASCIIData extends
 * FTPDataType.
 */
public class FTPASCIIData extends FTPDataType {

	/**
	 * 1.3.2. FTPASCIIData constructors.
	 */
	public FTPASCIIData() {
		super();
	}

	public FTPASCIIData(InetAddress server, int port) {
		super(server, port);
	}

	/**
	 * 3.1.2. ASCII and ASC replace their values setTxMode.
	 */
	/**
	 * 1.3.3. setTxMode method is overridden.
	 */
	public boolean setTxMode(String mode) {
		boolean modeIsValid = false;

		if (isASCIIMode(mode)) {
			modeIsValid = true;
			sendTxMode(mode);
		}

		if (modeIsValid)
			printOutputLine(readCtrl());

		return (modeIsValid);
	}

	/**
	 * 1.3.3. setTxMode method is overridden.
	 */
	public boolean setTxMode(char nm) {
		boolean rc = false;

		if (isASCIIMode(nm))
			rc = setTxMode("ascii");

		return (rc);
	}

	/**
```

```
        * 4.1. Extract Method: sendTxMode method.
        */
       protected void sendTxMode(String mode) {
              if (isASCIIMode(mode)) {
                     send("type a");
              }
       }

       /**
        * 4.1. Extract Method: printOutputLine methods.
        */
       /**
        * 4.4. Pull Up Method: printOutputLine method is pulled up to
        * FTPConnection.
        */
       /*
        * protected void printOutputLine(String message) {
        * getStdout().println(message); }
        */

       /**
        * 4.2. Decompose Conditional: isASCIIMode method with parameter of
String
        * type.
        */
       protected boolean isASCIIMode(String mode) {
              return ((mode.equals(ASCII)) || (mode.equals(ASC)));
       }

       /**
        * 4.2. Decompose Conditional: isASCIIMode method with parameter of
        * character type.
        */
       protected boolean isASCIIMode(char mode) {
              return ((mode == 'a'));
       }
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.io.IOException;
import java.net.InetAddress;

import jftplevelrefactoring.looplevel.interfaces.Authenticateable;

/**
 * 1.1.1. Extract Subclass: FTPAuthentication extends FTPCmdServer and
relevant
 * methods are pushed down. 1.2.1. FTPAuthentication extends FTPConnection.
 */
public class FTPAuthentication extends FTPConnection implements
              Authenticateable {

       /**
        * 3.3.1. Push Down Field: userName, anonymousPasswd and authenticated
are
```

```
         * pushed down with encapsulation methods from JFTPSuper.
         */
        private String userName = "me";
        private String anonymousPasswd = "tux@northpole.org";
        private boolean authenticated = false;

        /**
         * 1.1.2. FTPAuthentication constructors.
         */
        public FTPAuthentication() {
               super();
        }

        public FTPAuthentication(InetAddress server, int port) {
               super(server, port);
        }

        public FTPAuthentication(String userName, String anonymousPasswd,
                      boolean authenticated) {
               super();
               this.userName = userName;
               this.anonymousPasswd = anonymousPasswd;
               this.authenticated = authenticated;
        }

        public   FTPAuthentication(InetAddress   server,   int   port,   String
userName,
                      String anonymousPasswd, boolean authenticated) {
               super(server, port);
               this.userName = userName;
               this.anonymousPasswd = anonymousPasswd;
               this.authenticated = authenticated;
        }

        /**
         *  3.3.2.  Encapsulation  methods  are  pushed  down  with  fields  from
JFTPSuper.
         */

        /**
         * Returns the userName
         *
         * @return The current userName
         *@since 0.0.2pA2
         */
        public String getUserName() {
               return (userName);
        }

        /**
         * Sets the userName
         *
         * @param s
         *             New userName
         *@since 0.0.2pA2
         */
        public void setUserName(String s) {
               userName = s;
        }

        /**
         * Returns the anonymous passwd.
         *
```

```
	 * @return The password used for anonymous login.
	 *@since 0.0.2pA2
	 */
	public String getAnonymousPasswd() {
		return (anonymousPasswd);
	}

	/**
	 * Sets the password used for anonymous login to s.
	 *
	 * @param s
	 *            The password used for anonymous login.
	 *@since 0.0.2pA2
	 */
	public void setAnonymousPasswdTo(String s) {
		anonymousPasswd = s;
	}

	/**
	 * Are we authenticated ?
	 *
	 * @return true if we are authenticated; else false
	 *@since 0.0.2pA2
	 */
	public boolean isAuthenticated() {
		return (authenticated);
	}

	/**
	 * Sets authenticated to true
	 *
	 * @since 0.0.2pA2
	 */
	public void setAuthenticated() {
		authenticated = true;
	}

	/**
	 * Sets authenticated to false
	 *
	 * @since 0.0.2pA2
	 */
	public void unsetAuthenticated() {
		authenticated = false;
	}

	/**
	 * 1.1.3. Relevant methods are pushed down from FTPCmdServer.
	 */

	/* <START Implementing "Authenticateable"> */

	/**
	 * Authenticates an user
	 *
	 * @return true if user is authenticated; otherwise false
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public boolean authenticate() {
		boolean rc = true;
		String passwd = "", strReceive;
```

```
            try {
                  unsetAuthenticated();
                  printOutput("Username: ");
                  setUserName(readInputLine());  //  shouldn't  we  check
this?

                  send("user " + getUserName());
                  strReceive = readCtrl();
                  printOutputLine(strReceive);
                  if (usernameIsOkay(strReceive)) {
                        if (getUserName().equals("anonymous")) {
                              passwd = getAnonymousPasswd();
                        } else {
                              printOutput("Password: ");
                        }

                        try {
                              passwd = readInputPassword();
                        } catch (IOException ioe) {
                              printErrorLine("Unable    to    read    the
password.");
                        }

                        send("pass " + passwd);

                        if (getDebugLevel() >= 2)
                              printErrorLine("User: " + getUserName() + "
Pwd: " + passwd);

                        strReceive = this.readCtrl();
                        if (!(userIsLoggedIn(strReceive))) {
                              rc = false;
                        }
                        printOutputLine(strReceive);
                  } else {
                        printErrorLine("Login failed.");
                        rc = false;
                  }
            } catch (Exception e) {
                  rc = false;
                  //   printErrorLine("Unable   to   authenticate.");//Only
debug(tk)
            }

            return (rc);
      }

      /* <END Implementing "Authenticateable"> */

      /**
       *    4.1.    Extract    Method:    readInputLine,    readInputPassword,
printOutputLine,
       * printOutput and printErrorLine methods.
       */
      /**
       *   4.4.   Pull   Up   Method:   readInputLine,   readInputPassword,
printOutputLine,
       *  printOutput  and  printErrorLine  methods  are  pulled  up  to
FTPConnection.
       */
      /*
       * protected String readInputLine() throws IOException { return
       * getStdin().readLine(); }
```

```
         *
         * protected String readInputPassword() throws IOException { return
         * getStdin().readPasswd(); }
         *
         * protected void printOutputLine(String message) {
         * getStdout().println(message); }
         *
         *       protected       void       printOutput(String       message)
{ getStdout().print(message);
         * }
         *
         * protected void printErrorLine(String message) {
         * getStderr().println(message); }
         */

        /**
         * 4.2. Decompose Conditional: usernameIsOkay method.
         */
        private boolean usernameIsOkay(String response) {
                return (response.startsWith("331"));
        }

        /**
         * 4.2. Decompose Conditional: userIsLoggedIn method.
         */
        private boolean userIsLoggedIn(String response) {
                return (response.startsWith("230"));
        }
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.net.InetAddress;

/**
 * 1.3.1. Replace Conditional with Polymorphism: FTPBinaryData extends
 * FTPDataType.
 */
public class FTPBinaryData extends FTPDataType {

        /**
         * 1.3.2. FTPBinaryData constructors.
         */
        public FTPBinaryData() {
                super();
        }

        public FTPBinaryData(InetAddress server, int port) {
                super(server, port);
        }

        /**
         * 3.1.2. BINARY and BIN replace their values setTxMode.
         */
        /**
         * 1.3.3. setTxMode method is overridden.
```

```
         */
        public boolean setTxMode(String mode) {
                boolean modeIsValid = false;

                if (isBinaryMode(mode)) {
                        modeIsValid = true;
                        sendTxMode(mode);
                }

                if (modeIsValid)
                        printOutputLine(readCtrl());

                return (modeIsValid);
        }

        /**
         * 1.3.3. setTxMode method is overridden.
         */
        public boolean setTxMode(char nm) {
                boolean rc = false;

                if (isBinaryMode(nm))
                        rc = setTxMode("binary");

                return (rc);
        }

        /**
         * 4.1. Extract Method: sendTxMode method.
         */
        protected void sendTxMode(String mode) {
                if (isBinaryMode(mode)) {
                        send("type i");
                }
        }

        /**
         * 4.1. Extract Method: printOutputLine methods.
         */
        /**
         * 4.4. Pull Up Method: printOutputLine method is pulled up to
         * FTPConnection.
         */
        /*
         * protected void printOutputLine(String message) {
         * getStdout().println(message); }
         */

        /**
         * 4.2. Decompose Conditional: isBinaryMode method with parameter of
String
         * type.
         */
        protected boolean isBinaryMode(String mode) {
                return ((mode.equals(BINARY)) || (mode.equals(BIN)));
        }

        /**
         * 4.2. Decompose Conditional: isBinaryMode method with parameter of
         * character type.
         */
        protected boolean isBinaryMode(char mode) {
                return ((mode == 'i') || (mode == 'b'));
```

```
        }
}

/*
 * Name:    FTPCmdServer.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name               changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz       Creation
 * 0.0.2pA2 02.05.2001 Svenja Wittstadt   authenticate method implemented
 * 0.0.2pA3 06.05.2001 Tobias Kranz       added skeleton of 'int getPort()'.
 * 0.0.2pA4 15.05.2001 Tobias Kranz       changed 'String receiveLine()' &
 *                                        'String receiveBlock()'.
 * 0.0.2pA4 28.05.2001 Tobias Kranz       some small fixes...
 * 0.0.2pA4 06.06.2001 Tobias Kranz       REWRITE from scratch
 * 0.1.0    07.06.2001 Tobias Kranz &&
 *                     Sebastian Schipper Completed the rewrite.
 *                                        EVERYTHING´s FINE! ;-]
 * 0.1.0    09.06.2001 Martin Loh         Implemented the putFile Method and
 *                                        updated the getFile method
 * 0.1.99    10.06.2001 Tobias Kranz          Fixed bugs: cantListAfterPut,
cantGet.
 *                                        add 'setTxMode(String)' &
 *                                        'getFileSize(String)'
 * 0.1.99t1 11.06.2001 Tobias Kranz         added Progressbar in 'putFile()'
&&
 *                                        'getFile()'.
 * 0.1.99t1 12.06.2001 Tobias Kranz       added 'getFiles(String regex)'
 *                                            'not sure if this is the right
place..
 * 0.1.99t1 19.06.2001 Tobias Kranz       improved 'getFileSize(String)' I
 * 0.1.99t2 17.07.2001 Tobias Kranz       improved 'getFileSize(String)' II
 *                                        added 'getDirListArray()' &&
 *                                        'getLocalDirListArray()' &&
 *                                        'getLocalLongDirList()' &&
 *                                        'getLocalShortDirList()'
 * 0.1.99t4 19.07.2001 Tobias Kranz       Made an improvement suggestion in
 *                                                 'getFile()' && started to
implement
 *                                                   the "Createable" && the
"Removeable"
 *                                        interfaces.
 * 0.1.99t5 23.07.2001 Tobias Kranz        changed 'getFile()' && 'putFile()'
to
 *                                        work in the local working dir
 * 0.1.99t5 01.08.2001 Tobias Kranz       changed 'authenticate()'
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;

import jftplevelrefactoring.looplevel.io.NetReader;
import jftplevelrefactoring.looplevel.io.NetWriter;
import jftplevelrefactoring.looplevel.io.StdErr;
import jftplevelrefactoring.looplevel.io.StdIn;
import jftplevelrefactoring.looplevel.io.StdOut;
```

```
/**
 * This class contains the implementation of all supported FTP-Commands
 *
 * @since 0.0.2pA1
 */
/**
 * 1.2.0. Remove final keyword from FTPCmdServer.
 */
public class FTPCmdServer extends JFTPSuper {
	private Socket ctrlSock;
	private Socket dataSock;

	private NetReader ctrlReader;
	private NetWriter ctrlWriter;
	private NetReader dataReader;
	private NetWriter dataWriter;

	private InputStream dataIs;
	private OutputStream dataOs;

	private InputStreamReader ctrlIsr;
	private InputStreamReader dataIsr;

	private StdIn stdin = new StdIn();
	private StdOut stdout = new StdOut();
	private StdErr stderr = new StdErr();

	/**
	 * 3.2. Encapsulate Fields: getCtrlSock and setCtrlSock.
	 */

	/**
	 * @return the ctrlSock
	 */
	public Socket getCtrlSock() {
		return ctrlSock;
	}

	/**
	 * @param ctrlSock
	 *            the ctrlSock to set
	 */
	public void setCtrlSock(Socket ctrlSock) {
		this.ctrlSock = ctrlSock;
	}

	/**
	 * 3.2. Encapsulate Fields: getDataSock and setDataSock.
	 */

	/**
	 * @return the dataSock
	 */
	public Socket getDataSock() {
		return dataSock;
	}

	/**
	 * @param dataSock
	 *            the dataSock to set
	 */
	public void setDataSock(Socket dataSock) {
```

```
		this.dataSock = dataSock;
	}

	/**
	 * 3.2. Encapsulate Fields: getCtrlReader and setCtrlReader.
	 */

	/**
	 * @return the ctrlReader
	 */
	public NetReader getCtrlReader() {
		return ctrlReader;
	}

	/**
	 * @param ctrlReader
	 *            the ctrlReader to set
	 */
	public void setCtrlReader(NetReader ctrlReader) {
		this.ctrlReader = ctrlReader;
	}

	/**
	 * 3.2. Encapsulate Fields: getCtrlWriter and setCtrlWriter.
	 */

	/**
	 * @return the ctrlWriter
	 */
	public NetWriter getCtrlWriter() {
		return ctrlWriter;
	}

	/**
	 * @param ctrlWriter
	 *            the ctrlWriter to set
	 */
	public void setCtrlWriter(NetWriter ctrlWriter) {
		this.ctrlWriter = ctrlWriter;
	}

	/**
	 * 3.2. Encapsulate Fields: getDataReader and setDataReader.
	 */

	/**
	 * @return the dataReader
	 */
	public NetReader getDataReader() {
		return dataReader;
	}

	/**
	 * @param dataReader
	 *            the dataReader to set
	 */
	public void setDataReader(NetReader dataReader) {
		this.dataReader = dataReader;
	}

	/**
	 * 3.2. Encapsulate Fields: getDataWriter and setDataWriter.
	 */
```

```
	/**
	 * @return the dataWriter
	 */
	public NetWriter getDataWriter() {
		return dataWriter;
	}

	/**
	 * @param dataWriter
	 *            the dataWriter to set
	 */
	public void setDataWriter(NetWriter dataWriter) {
		this.dataWriter = dataWriter;
	}

	/**
	 * 3.2. Encapsulate Fields: getDataIs and setDataIs.
	 */

	/**
	 * @return the dataIs
	 */
	public InputStream getDataIs() {
		return dataIs;
	}

	/**
	 * @param dataIs
	 *            the dataIs to set
	 */
	public void setDataIs(InputStream dataIs) {
		this.dataIs = dataIs;
	}

	/**
	 * 3.2. Encapsulate Fields: getDataOs and setDataOs.
	 */

	/**
	 * @return the dataOs
	 */
	public OutputStream getDataOs() {
		return dataOs;
	}

	/**
	 * @param dataOs
	 *            the dataOs to set
	 */
	public void setDataOs(OutputStream dataOs) {
		this.dataOs = dataOs;
	}

	/**
	 * 3.2. Encapsulate Fields: getCtrlIsr and setCtrlIsr.
	 */

	/**
	 * @return the ctrlIsr
	 */
	public InputStreamReader getCtrlIsr() {
		return ctrlIsr;
```

```
	}

	/**
	 * @param ctrlIsr
	 *            the ctrlIsr to set
	 */
	public void setCtrlIsr(InputStreamReader ctrlIsr) {
		this.ctrlIsr = ctrlIsr;
	}

	/**
	 * 3.2. Encapsulate Fields: getDataIsr and setDataIsr.
	 */

	/**
	 * @return the dataIsr
	 */
	public InputStreamReader getDataIsr() {
		return dataIsr;
	}

	/**
	 * @param dataIsr
	 *            the dataIsr to set
	 */
	public void setDataIsr(InputStreamReader dataIsr) {
		this.dataIsr = dataIsr;
	}

	/**
	 * 3.2. Encapsulate Fields: getStdin and setStdin.
	 */

	/**
	 * @return the stdin
	 */
	public StdIn getStdin() {
		return stdin;
	}

	/**
	 * 4.3. Remove Setting Method: setStdin method is removed because it
is set
	 * at creation time and never altered.
	 */
	/**
	 * @param stdin
	 *            the stdin to set
	 */
	/*
	 * public void setStdin(StdIn stdin) { this.stdin = stdin; }
	 */

	/**
	 * 3.2. Encapsulate Fields: getStdout and setStdout.
	 */

	/**
	 * @return the stdout
	 */
	public StdOut getStdout() {
		return stdout;
	}
```

```
	/**
	 * 4.3. Remove Setting Method: setStdout method is removed because it
is set
	 * at creation time and never altered.
	 */
	/**
	 * @param stdout
	 *            the stdout to set
	 */
	/*
	 * public void setStdout(StdOut stdout) { this.stdout = stdout; }
	 */

	/**
	 * 3.2. Encapsulate Fields: getStderr and setStderr.
	 */

	/**
	 * @return the stderr
	 */
	public StdErr getStderr() {
		return stderr;
	}

	/**
	 * 4.3. Remove Setting Method: setStderr method is removed because it
is set
	 * at creation time and never altered.
	 */
	/**
	 * @param stderr
	 *            the stderr to set
	 */
	/*
	 * public void setStderr(StdErr stderr) { this.stderr = stderr; }
	 */
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.io.IOException;
import java.net.BindException;
import java.net.ConnectException;
import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.NoRouteToHostException;
import java.net.ProtocolException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.net.UnknownServiceException;
import java.util.BitSet;
import java.util.StringTokenizer;

import jftplevelrefactoring.looplevel.interfaces.Connectable;
import jftplevelrefactoring.looplevel.io.NetReader;
```

```
import jftplevelrefactoring.looplevel.io.NetWriter;
import jftplevelrefactoring.looplevel.io.StdErr;

/**
 * 1.1.1. Extract Subclass: FTPConnection extends FTPCmdServer and relevant
 * methods are pushed down. 1.2.1. Extract Super-Class: FTPConnection is set
to
 * be super class for other FTPCmdServer subclasses.
 */
public class FTPConnection extends FTPCmdServer implements Connectable {

	/**
	 * 3.3.1. Push Down Field: server and port are pushed down with
	 * encapsulation methods from JFTPSuper.
	 */
	private InetAddress server;
	private int port = 21;

	/**
	 * 1.3.1. Push Down Field: connected and passiveConnected options are
pushed
	 * down with encapsulation methods from JFTPSuper.
	 */
	private BitSet options = new BitSet(2);

	/**
	 * 1.1.2. FTPConnection constructors.
	 */
	public FTPConnection() {
		super();
		try {
			this.server = InetAddress.getByName("localhost");
		} catch (UnknownHostException uhe) {
			System.err.println("COUGHT!");
		}
		/*
		 * BitSet 'options'
		 *
		 *  #:  if  true:  default:  0  =>  connected  false,  1  =>
passiveConnected
		 * false.
		 */
		this.options.clear(0);
		this.options.clear(1);
	}

	public FTPConnection(InetAddress server, int port) {
		super();
		this.server = server;
		this.port = port;
	}

	/**
	 *  3.3.2.  Encapsulation  methods  are  pushed  down  with  fields  from
JFTPSuper.
	 */

	/**
	 * Returns the current Server
	 *
	 * @return InetAddress of the Server
	 *@since 0.0.2pA1
	 */
```

```
	public InetAddress getServer() {
		return (server);
	}

	/**
	 * Returns the IP of the Server
	 *
	 * @return the servers IP
	 *@since 0.0.2pA1
	 */
	public String getServerIP() {
		return (server.getHostAddress());
	}

	/**
	 * Returns the Name of the Server
	 *
	 * @return the servers Name
	 *@since 0.0.2pA1
	 */
	public String getServerName() {
		return (server.getHostName());
	}

	/**
	 * sets the Servers IP/Name
	 *
	 * @param s
	 *            Servers name/IP
	 *@return true if the Server can be set; otherwise false
	 *@version 2
	 *@since 0.0.2pA1
	 */
	public boolean setServerTo(String s) {
		boolean rc;

		try {
			server = InetAddress.getByName(s);

			rc = true;
		} catch (UnknownHostException uhe) {
			StdErr stderr = new StdErr();
			stderr.println("Unable to reach host '" + s + "'.");
			rc = false;
		}

		return (rc);
	}

	/**
	 * What's the current port?
	 *
	 * @return The current port
	 *@since 0.0.2pA1
	 */
	public int getServerPort() {
		return (port);
	}

	/**
	 * Sets the port
	 *
	 * @param port
```

```
         *            Port
         *@since 0.0.2pA1
         */
        public void setServerPortTo(int port) {
               this.port = port;
        }

        /**
         *  3.3.2.  Encapsulation  methods  are  pushed  down  with  fields  from
JFTPSuper.
         */

        /**
         * Are we connected to a ftpserver ?
         *
         * @return true if connected; otherwise false
         *@since 0.0.2pA1
         */
        public boolean isConnected() {
               return ((boolean) options.get(0));
        }

        /**
         * Sets connected to true.
         *
         * @since 0.0.2pA1
         */
        protected void setConnected() {
               options.set(0);
        }

        /**
         * Sets connected to false.
         *
         * @since 0.0.2pA1
         */
        protected void unsetConnected() {
               options.clear(0);
        }

        /**
         * Are we passive Connected
         *
         * @return true if we are passive connected; otherwise false
         *@since 0.0.2pA4
         *@version 0
         */
        protected boolean isPassiveConnected() {
               return (options.get(1));
        }

        /**
         * Sets passive connected to true
         *
         * @since 0.0.2pA4
         *@version 1
         */
        protected void setPassiveConnected() {
               options.set(1);
        }

        /**
         * Sets passive connected to false
```

```
         *
         * @since 0.0.2pA4
         *@version 1
         */
        protected void unsetPassiveConnected() {
                options.clear(1);
        }

        /**
         * 1.1.3. Relevant methods are pushed down from FTPCmdServer.
         */

        /* <START Implementing "Connectable"> */

        /**
         * Opens a connection
         *
         * @return true if connected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean connect() {
                // return (openCtrlConnection());
                return (connect('c'));
        }

        /**
         * 4.7.1. Parameterize Method: connect method is overloaded by
         *  parameterizing  it  with  connection  type  flag  where  c  represents
control
         *  connection  and  d  represents  passive  data  connection.  4.7.2.
Parameterized
         * connect method is invoked in original connect method.
         */
        public boolean connect(char type) {
                if (type == 'c')
                        return (openCtrlConnection());
                else if (type == 'd')
                        return (openPassiveDataConnection());
                else
                        return false;
        }

        /**
         * Opens a ctrl connection
         *
         * @return true if connected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean openCtrlConnection() {
                boolean rc = false;

                if (prepareCtrlConnection()) {
                        if (getDebugLevel() >= 2)
                                printErrorLine("Trying  to  get  Streams  from  the
ctrlSock.");

                        this.setCtrlReader(new NetReader(getCtrlSock()));
                        this.setCtrlWriter(new NetWriter(getCtrlSock()));

                        setConnected();
                        rc = true;
```

```
            } else {
                  printErrorLine("Unable to connect.");
            }

            return (rc);
      }

      /**
       * Closes a connection
       *
       * @return true if disconnected; false if not
       *@since 0.1.0
       *@version 0
       */
      public boolean disconnect() {
            // return (closeCtrlConnection());
            return (disconnect('c'));
      }

      /**
       * 4.7.1. Parameterize Method: disconnect method is overloaded by
       * parameterizing it with connection type flag where c represents
control
       * connection and d represents passive data connection. 4.7.2.
Parameterized
       * disconnect method is invoked in original disconnect method.
       */
      public boolean disconnect(char type) {
            if (type == 'c')
                  return (closeCtrlConnection());
            else if (type == 'd')
                  return (closePassiveDataConnection());
            else
                  return false;
      }

      /**
       * Closes a ctrl connection
       *
       * @return true if disconnected; false if not
       *@since 0.1.0
       *@version 0
       */
      public boolean closeCtrlConnection() {
            boolean rc = false;

            if (isConnected()) {
                  send("QuiT");
                  printOutputLine(readCtrl());

                  try {
                        getCtrlSock().close();
                        setCtrlSock(null);
                        setCtrlIsr(null);
                        setCtrlReader(null);
                        setCtrlWriter(null);

                        rc = true;
                  } catch (IOException ioe) {
                        writeLineToConsole("Exception    while    closing
ControlSocket.");
                  }
```

```
                        unsetConnected();
                }

                return (rc);
        }

        /**
         * Opens a passive connection for data.
         *
         * @return true if connected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean openPassiveDataConnection() {
                int port;
                boolean rc = true;
                String receive;

                if (preparePassiveDataConnection()) {
                        unsetPassiveConnected();

                        send("pasv");
                        receive = readCtrl();
                        port = getPort(receive);

                        if (port == -1) {
                                rc = false;
                        } else {
                                if (bindDataSocket()) {
                                        this.setDataReader(new
NetReader(getDataSock()));
                                        this.setDataWriter(new
NetWriter(getDataSock()));

                                        setPassiveConnected();
                                } else // bindDataSocket(..) FAILED
                                {
                                        rc = false;
                                }
                        }
                }

                return (rc);
        }

        /**
         * Closes a passive connection for data.
         *
         * @return true if disconnected; false if not
         *@since 0.1.0
         *@version 0
         */
        public boolean closePassiveDataConnection() {
                boolean rc = false;

                if (isPassiveConnected()) {
                        try {
                                getDataSock().close();
                                setDataSock(null);
                                setDataIs(null);
                                setDataOs(null);
                                setDataIsr(null);
                                setDataReader(null);
```

```
                        setDataWriter(null);

                        rc = true;
                    } catch (IOException ioe) {
                        writeLineToConsole("E...while             closeing
dataSock.");
                    }
            }

            unsetPassiveConnected();

            return (rc);
      }

      /* <START Implementing tools for "Connectable"> */
      /**
       * Binds a CtrlSocket to the Host/Port - pair set in JFTPSuper
       *
       * @return true if socket is bind; else false
       *@since 0.1.0
       *@version 0
       */
      private boolean bindCtrlSocket() {
            return (bindSocket('c'));
      }

      /**
       *   4.5.   Remove   Parameter:   port   parameter   is   removed   from
bindDataSocket
       * method by using port instance variable.
       */
      /**
       * Binds a DataSocket to the Host/Port - pair set in JFTPSuper
       *
       * @return true if socket is bind; else false.
       *@since 0.1.0
       *@version 0
       */
      private boolean bindDataSocket() {
            return (bindSocket('d'));
      }

      /**
       * 4.5. Remove Parameter: server and port parameters are removed from
       * bindSocket method by using server and port instance variables.
       */
      /**
       * Binds a Socket to the given Host/Port - pair
       *
       * @param sock
       *            The type of socket (only 'c' or 'd' are valid).
       *@param server
       *            The server to bind to.
       *@param port
       *            The Port to bind to.
       *@return true if socket is bind; else false
       *@since 0.1.0
       *@version 0
       */
      private boolean bindSocket(char sock) {
            boolean rc = false;

            if (getDebugLevel() >= 2)
```

```
                    printErrorLine("Binding   Socket   to   server   "   +
getServerName() + ":"
                                + getServerPort());
             /* NO! ^ Not the beer */

             try {
                    if (sock == 'd')
                           setDataSock(new            Socket(getServerName(),
getServerPort()));
                    else if (sock == 'c')
                           setCtrlSock(new            Socket(getServerName(),
getServerPort()));
                    else
                           printErrorLine("Ooops... You should __NEVER__ get
here! hum?");

                    rc = true;
             } catch (BindException be) {
                    printErrorLine("Can't bind socket.");
             } catch (MalformedURLException mue) {
                    printErrorLine("Malformed URL.");
             } catch (ConnectException ce) {
                    printErrorLine("Connection refused by Server.");
             } catch (NoRouteToHostException nrthe) {
                    printErrorLine("No route to host.");
             } catch (ProtocolException pe) {
                    printErrorLine("A protocol error occured.");
             } catch (UnknownHostException uhe) {
                    printErrorLine("Unable   to   resolve   IP-address   of
specified host.");
             } catch (UnknownServiceException use) {
                    printErrorLine("Used protocol/service is not known.");
             } catch (IOException ioe) {
                    printErrorLine("Unknown IO-Error.");
             }

             if (getDebugLevel() >= 2)
                    printErrorLine("Socket is bound (" + rc + ")");

             return (rc);
       }

       /**
        * 4.8. Remove Assignments to Parameters: assignments of parameter of
        * getPort method are removed by using temporary variable.
        */
       /**
        * Returns the port to use for PASV
        *
        * @param s
        *            String to parse for the Port
        *@return the port to use; or -1 if it can't resolve the port.
        *@since 0.0.2pA1
        */
       private int getPort(String s) {
              /*
               * Now get the port to connect to.
               *
               * To do this we have to parse a string like this:
               * "227 Entering Passive Mode (10,1,1,1,4,13)" where '10,1,1,1'
is the
               * IP-Ad. and '4,13' stands for the port. ('4' are the first
8bits of an
```

```
                * 16bit long number and '13' is the second.) In this case the
port
                * would be 1037 (4256+13).
                *
                *   At   first   we   are   creating   a   substring   containing
"(10,1,1,1,4,13)" to
                * parse and get the 5th and 6th Token, convert them to int and
                * calculate the _REAL_ port to give it back.
                */
               int rc = -1, port1, port2, i;
               StringTokenizer st;

               /**
                * 4.8.1. Temporary variable is set value of parameter. 4.8.2.
All
                * assignments to parameter are set to temporary variable.
                */
               String temp = s;
               try {
                     temp          =          temp.substring(temp.indexOf('('),
temp.indexOf(')'));

                     st = new StringTokenizer(temp, ",", false);

                     if (st.countTokens() == 6) {
                            for (i = 0; i < 4; i++)
                                  st.nextToken();

                            port1 = Integer.parseInt(st.nextToken());
                            port2 = Integer.parseInt(st.nextToken());
                            /*
                             * Now "calculate" them this way:
                             *
                             * 01010011 . 10101100 = 0101001110101100 ^ ^ ^ ^
8bit . 8bit =
                             * 16bit ^ ^ ^ ^ 1st port concatenation 2nd port =
_REAL_ port
                             */
                            rc = (port1 * 256) + port2;
                     }
               } catch (Exception e) {
                     rc = -1;
               }

               return (rc);
        }

        /* <END Implementing tools for "Connectable"> */
        /* <END Implementing "Connectable"> */

        /**
         * 1.4. Hide Delegate: FTPTransfer is hidden by readCtrl and send
methods.
         */
        protected String readCtrl() {
               FTPTransfer transfer = new FTPTransfer(this.getServer(), this
                            .getServerPort());
               return transfer.readCtrl();
        }

        protected void send(String s) {
               FTPTransfer transfer = new FTPTransfer(this.getServer(), this
                            .getServerPort());
```

```
                transfer.send(s);
        }

        /**
         * 4.1. Extract Method: printOutputLine, printErrorLine and
         * writeLineToConsole methods.
         */
        protected void printOutputLine(String message) {
                getStdout().println(message);
        }

        protected void printErrorLine(String message) {
                getStderr().println(message);
        }

        protected void writeLineToConsole(String message) {
                System.out.println(message);
        }

        /**
         *   4.4.   Pull   Up   Method:   printOutput,   readInputLine   and
readInputPassword
         * methods are pulled up to FTPConnection.
         */
        protected void printOutput(String message) {
                getStdout().print(message);
        }

        protected String readInputLine() throws IOException {
                return getStdin().readLine();
        }

        protected String readInputPassword() throws IOException {
                return getStdin().readPasswd();
        }

        /**
         * 4.2. Decompose Conditional: preparePassiveDataConnection method.
         */
        private boolean preparePassiveDataConnection() {
                return ((isConnected()) && (!isPassiveConnected()));
        }

        /**
         * 4.2. Decompose Conditional: prepareCtrlConnection method.
         */
        private boolean prepareCtrlConnection() {
                return ((!isConnected()) && (bindCtrlSocket()));
        }
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.net.InetAddress;

import jftplevelrefactoring.looplevel.interfaces.DataTypeChangeable;
```

```
/**
 * 1.1.1. Extract Subclass: FTPDataType extends FTPCmdServer and relevant
 * methods are pushed down. 1.2.1. FTPDataType extends FTPConnection. 1.3.4.
 * FTPDataType is converted to abstract class.
 */
public abstract class FTPDataType extends FTPConnection implements
            DataTypeChangeable {

      /**
       * 3.1.1. Replace Magic Number with Symbolic Constant: ASCII, ASC,
BINARY
       * and BIN.
       */
      protected String ASCII = "ascii";
      protected String ASC = "asc";
      protected String BINARY = "binary";
      protected String BIN = "bin";

      /**
       * 1.1.2. FTPDataType constructors.
       */
      public FTPDataType() {
            super();
      }

      public FTPDataType(InetAddress server, int port) {
            super(server, port);
      }

      /**
       * 1.1.3. Relevant methods are pushed down from FTPCmdServer.
       */

      /* <START Implementing "Changeable"> */

      /**
       * 3.1.2. ASCII, ASC, BINARY and BIN replace their values setTxMode.
       */

      /**
       * 2.3.5. setTxMode method is converted to abstract method.
       */

      /**
       * Sets the transfermode either to ascii or to binary
       *
       * @param mode
       *            The mode to use ("ascii" or "binary")
       *@return true if mode is set, otherwise false
       *@since 0.1.99
       *@version 0
       */
      public abstract boolean setTxMode(String mode);

      /**
       * 3.1.2. ASCII, and BINARY replace their values setTxMode.
       */

      /**
       * 1.3.5. setTxMode method is converted to abstract method.
       */
      public abstract boolean setTxMode(char nm);
```

```
	/* <END Implementing "Changeable"> */
	protected abstract void sendTxMode(String mode);
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.io.File;
import java.net.InetAddress;
import java.util.StringTokenizer;

import jftplevelrefactoring.looplevel.interfaces.Createable;
import jftplevelrefactoring.looplevel.interfaces.DirectoryChangeable;
import jftplevelrefactoring.looplevel.interfaces.Listable;
import jftplevelrefactoring.looplevel.interfaces.Removeable;

/**
 * 1.1.1. Extract Subclass: FTPDirectory extends FTPCmdServer and relevant
 * methods are pushed down. 1.1.2. FTPDirectory extends FTPConnection.
 */
public class FTPDirectory extends FTPConnection implements Createable,
		DirectoryChangeable, Listable, Removeable {

	/**
	 * 3.3.1. Push Down Field: currentWorkingDir is pushed down with
	 * encapsulation methods from JFTPSuper.
	 */
	private File currentWorkingDir = new File(".");

	/**
	 * 1.1.2. FTPDirectory constructors.
	 */
	public FTPDirectory() {
		super();
	}

	public FTPDirectory(InetAddress server, int port) {
		super(server, port);
	}

	public FTPDirectory(File currentWorkingDir) {
		super();
		this.currentWorkingDir = currentWorkingDir;
	}

	public FTPDirectory(InetAddress server, int port, File
currentWorkingDir) {
		super(server, port);
		this.currentWorkingDir = currentWorkingDir;
	}

	/**
	 * 3.3.2. Encapsulation methods are pushed down with fields from
JFTPSuper.
	 */
```

```
	/**
	 * Returns a java.lang.String containing the current working directory
	 *
	 * @return A java.lang.String
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String getCurrentWorkingDir() {
		String rc = currentWorkingDir.getAbsolutePath();
		rc = rc.substring(0, rc.length() - 1);

		return (rc);
	}

	/**
	 * 5.3. Reverse Conditional: setCurrentWorkingDir method.
	 */
	/**
	 * Sets the current working dir to the given java.lang.String
	 *
	 * @param A
	 *            java.lang.String
	 *@return true if path is set; otherwise false
	 *@since 0.1.99t4
	 *@version 2
	 */
	/*
	 * Some infos: Java's path-format is
	 *  'DIR_SEPARATOR''PATH''DIR_SEPARATOR''.'. eg: / PATH / . . eg:
"/root/."
	 * || "C:\MICROS~1\."
	 */
	public boolean setCurrentWorkingDir(String newPath) {
		boolean rc = true;
		String cp;

		if (isAbsolutePath(newPath)) // absolute
		{

			/**
			 *  5.3. Reverse Conditional: existing absolute path
logic.
			 */
			/*
			 * if (!(new File(newPath)).exists()) rc = false; else
			 * currentWorkingDir = new File(newPath + File.separator
+ ".");
			 */
			if ((new File(newPath)).exists())
				currentWorkingDir = new File(newPath +
File.separator + ".");
			else
				rc = false;
		} else if (isDirectoryUp(newPath)) // Dir up
		{
			cp = getCurrentWorkingDir();
			int lastIndex = cp.length() - 2; // cutting '\n' from
the end
			StringTokenizer st = new StringTokenizer(newPath,
File.separator);

			do {
				lastIndex = cp.lastIndexOf("/", lastIndex);
```

```
                        cp = cp.substring(0, lastIndex);
                        // pop ".." to /dev/null (we just need the 'st' as
counter)
                        st.nextToken();
                } while ((st.hasMoreTokens()));

                cp += File.separator + ".";

                if (!(new File(cp)).exists())
                        rc = false;
                else
                        currentWorkingDir = new File(cp);
            } else // relative
            {
                cp = getCurrentWorkingDir();
                if (isRelativePath(cp))
                        cp = cp.substring(0, cp.length() - 1);

                /**
                 *  5.3.  Reverse  Conditional:  existing  relative  path
logic.
                 */
                /*
                 *  if  (!(new  File(cp  +  newPath  +  File.separator  +
".")).exists()) rc
                 *  =  false;  else  currentWorkingDir  =  new  File(cp  +
newPath +
                 * File.separator + ".");
                 */
                if   ((new   File(cp   +   newPath   +   File.separator   +
".")).exists())
                        currentWorkingDir  =  new  File(cp  +  newPath  +
File.separator
                                        + ".");
                else
                        rc = false;
            }

            return (rc);
        }

        /**
         * 1.2.2. Relevant methods are pushed down from FTPCmdServer.
         */

        /* <START Implementing "Createable"> */

        /**
         * Creates a directory on the server named dirName
         *
         * @param dirName
         *            The name of the directory to create
         *@return true if successfully created; otherwise false
         *@since 0.1.99t4
         *@version 1
         */
        public boolean createRemoteDir(String dirName) {
            boolean rc = false; // senseless; but the compiler wants it ;-)
            String receive;

            send("mkd " + dirName.trim());
            receive = readCtrl();
            printOutputLine(receive);
```

```
		if (pathIsCreated(receive)) {
			rc = true;
		} else if (pathAlreadyExists(receive)) {
			rc = false;
		}

		return (rc);
	}

	/**
	 * Creates a directory on the localhost named dirName
	 *
	 * @param dirName
	 *            The name of the directory to create
	 *@return true if successfully created; otherwise false
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean createLocalDir(String dirName) {
		File newDir = new File(getCurrentWorkingDir() + File.separator
				+ dirName.trim());

		return (newDir.mkdir());
	}

	/* <END Implementing "Createable"> */

	/* <START Implementing "Changeable"> */
	public boolean changeRemoteWorkingDir(String newDir) {
		boolean rc = false;
		String receive;

		send("cwd " + newDir);
		receive = readCtrl();
		printOutputLine(receive);

		if (fileActionIsOkay(receive))
			rc = true;

		return (rc);
	}

	public boolean changeLocalWorkingDir(String newDir) {
		return (setCurrentWorkingDir(newDir));
	}

	/* <END Implementing "Changeable"> */

	/* <START Implementing "Listable"> */

	/**
	 * Return the current working directory on the server.
	 *
	 * @return A java.lang.String containing the working directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printRemoteWorkingDir() {
		send("pwd");
		return (readCtrl());
	}
```

```
	/**
	 * Return the current working directory on the localhost.
	 *
	 * @return A java.lang.String containing the working directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printLocalWorkingDir() {
		return (getCurrentWorkingDir());
	}

	/**
	 * Gives a detailed directorylist of the active directory.
	 *
	 * @return Directorylist of the active directory.
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public String getLongDirList() {
		String rc = "";

		if (openPassiveDataConnection()) {
			send("list");
			rc += readCtrl();
			if (!fileActionIsNotTaken(rc)) {
				rc += readData();
				rc += readCtrl();
			}
		} else {
			rc = "Unable to establish a DataConnection.";
		}

		if (isPassiveConnected())
			rc += "Unable to close the DataConnection.";

		return (rc);
	}

	/**
	 * 5.3. Reverse Conditional: getShortDirList method.
	 */
	/**
	 * Gives a short directorylist of the active directory.
	 *
	 * @return Directorylist of the active directory.
	 *@since 0.0.2pA1
	 *@version 1
	 */
	public String getShortDirList() {
		String rc = "";
		boolean con;

		if (!isPassiveConnected())
			con = openPassiveDataConnection();
		else
			con = true;

		if (con) {
			send("nlst");
			rc += readCtrl();

			/**
			 * 5.3. Reverse Conditional: not taken file action
```

```
logic.
                     */
                    /*
                     * if (!fileActionIsNotTaken(rc)) { rc += readData(); rc
+=
                     * readCtrl(); } else { closePassiveDataConnection(); }
                     */
                    if (fileActionIsNotTaken(rc)) {
                          closePassiveDataConnection();
                    } else {
                          rc += readData();
                          rc += readCtrl();
                    }
             } else {
                    rc = "Unable to establish a DataConnection.";
             }

             if (isPassiveConnected())
                    rc += "Unable to close the DataConnection.";

             return (rc);
       }

       /**
        * Returns a java.lang.String array containing the directorylist of
the
        * active directory.
        *
        * @return A java.lang.String array.
        *@since 0.1.99t2
        *@version 1
        */
       public String[] getDirListArray() {
             int i;
             StringTokenizer  st  =  new  StringTokenizer(getShortDirList(),
"\n\r");
             String[] rc = new String[st.countTokens() - 2];
             String tmp = "";

             st.nextToken(); // pop "150 Openi..." to /dev/null

             for (i = 0; i < rc.length; i++) {
                    tmp = st.nextToken();
                    rc[i] = tmp;
             }

             st.nextToken(); // pop "226 Trans..." to /dev/null

             if (getDebugLevel() >= 2)
                    printErrorLine("rc has # entries: " + rc.length);

             return (rc);
       }

       /**
        * Returns a short directorylist of the current working-directory
        *
        * @return A java.lang.String
        *@since 0.1.99t2
        *@version 1
        */
       public String getLocalShortDirList() {
             int i;
```

```
		String rc = "";
		File fi = new File(getCurrentWorkingDir());
		String[] tmp = fi.list();

		for (i = 0; i < tmp.length; i++) {
			rc += tmp[i] + "\n\r";
		}

		return (rc);
	}

	/**
	 *   6.1.   Split   Loop:   loop   of   files   list   properties   in
getLocalLongDirList
	 * method is duplicated to three separate loops as a loop per each
property
	 * (Directory, Readable and Writable).
	 */
	/**
	 * Returns a long directorylist of the current working-directory
	 *
	 * @return A java.lang.String
	 *@since 0.1.99.t2
	 *@version 1
	 */
	public String getLocalLongDirList() {
		String rc = "";
		String[] tmp;
		File[] fl;
		int i;
		File fi = new File(getCurrentWorkingDir());

		fl = fi.listFiles();
		tmp = new String[fl.length];

		/**
		 * 6.1.1. Extracted loop for Directory property of files list.
		 */
		for (i = 0; i < fl.length; i++) {
			// is Directory?
			if (fl[i].isDirectory())
				tmp[i] = "d";
			else
				tmp[i] = "-";

			// Size
			// tmp[i] += " " + fl[i].length();

			tmp[i] += " " + fl[i].getName();
		}

		/**
		 * 6.1.2. Extracted loop for Readable property of files list.
		 */
		for (i = 0; i < fl.length; i++) {
			// is Readable ?
			if (fl[i].canRead())
				tmp[i] += "r";
			else
				tmp[i] += "-";

			// Size
			// tmp[i] += " " + fl[i].length();
```

```
                        tmp[i] += " " + fl[i].getName();
                }

                /**
                 * 6.1.3. Extracted loop for Readable property of files list.
                 */
                for (i = 0; i < fl.length; i++) {
                        // is Writable ?
                        if (fl[i].canWrite())
                                tmp[i] += "w";
                        else
                                tmp[i] += "-";

                        // Size
                        // tmp[i] += " " + fl[i].length();

                        tmp[i] += " " + fl[i].getName();
                }

                // String[] -> String
                for (i = 0; i < tmp.length; i++) {
                        rc += tmp[i] + "\n\r";
                }

                return (rc);
        }

        /**
         * Returns a java.lang.String arry containing the directorylist of the
local
         * working-directory.
         *
         * @return A java.lang.String array.
         *@since 0.1.99t2
         *@version 1
         */
        /*
         * I know that this is not a really good place... ;-( (tk)
         */
        public String[] getLocalDirListArray() {
                File fi = new File(getCurrentWorkingDir());

                return (fi.list());
        }

        /* <END Implementing "Listable"> */

        /* <START Implementing "Removeable"> */

        /**
         * Removes a remote Directory called dirName
         *
         * @param dirName
         *            The name of the Directory to remove
         *@return true if successfully removed; otherwise false
         *@since 0.1.99t4
         *@version 1
         */
        public boolean removeRemoteDir(String dirName) {
                boolean rc = false; // senseless; but the compiler wants it ;-)
                String receive;
```

```
                send("rmd " + dirName.trim());
                receive = readCtrl();
                getStdout().println(receive);

                if (fileActionIsOkay(receive)) {
                        rc = true;
                } else if (fileActionIsNotTaken(receive)) {
                        rc = false;
                }

                return (rc);
        }

        /**
         * Removes a local Directory called dirName
         *
         * @param dirName
         *            The name of the Directory to remove
         *@return true if successfully removed; otherwise false
         *@since 0.1.99t4
         *@version 2
         */
        public boolean removeLocalDir(String dirName) {
                String cd = getCurrentWorkingDir();
                File rmDir = new File(cd.substring(0, cd.length() - 1) +
dirName.trim());

                return (rmDir.delete());
        }

        /* <END Implementing "Removeable"> */

        /**
         * 1.4. Hide Delegate: FTPTransfer is hidden by readData methods.
         */
        protected String readData() {
                FTPTransfer transfer = new FTPTransfer(this.getServer(), this
                                .getServerPort());
                return transfer.readData();
        }

        /**
         * 4.1. Extract Method: printOutputLine and printErrorLine methods.
         */
        /**
         * 4.4. Pull Up Method: printOutputLine and printErrorLine methods are
         * pulled up to FTPConnection.
         */
        /*
         * protected void printOutputLine(String message) {
         * getStdout().println(message); }
         *
         * protected void printErrorLine(String message) {
         * getStderr().println(message); }
         */

        /**
         * 4.2. Decompose Conditional: pathIsCreated method.
         */
        private boolean pathIsCreated(String response) {
                return (response.startsWith("257"));
        }
```

```
	/**
	 * 4.2. Decompose Conditional: pathAlreadyExists method.
	 */
	private boolean pathAlreadyExists(String response) {
		return (response.startsWith("521"));
	}

	/**
	 * 4.2. Decompose Conditional: fileActionIsOkay method.
	 */
	private boolean fileActionIsOkay(String response) {
		return (response.startsWith("250"));
	}

	/**
	 * 4.2. Decompose Conditional: actionIsNotTaken method.
	 */
	protected boolean fileActionIsNotTaken(String response) {
		return (response.startsWith("550"));
	}

	/**
	 * 4.2. Decompose Conditional: isAbsolutePath method.
	 */
	private boolean isAbsolutePath(String path) {
		return path.startsWith("/");
	}

	/**
	 * 4.2. Decompose Conditional: isDirectoryUp method.
	 */
	private boolean isDirectoryUp(String path) {
		return path.startsWith("..");
	}

	/**
	 * 4.2. Decompose Conditional: isRelativePath method.
	 */
	private boolean isRelativePath(String path) {
		return path.endsWith(".");
	}
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.InetAddress;
import java.util.StringTokenizer;
```

```
import jftplevelrefactoring.looplevel.interfaces.Transferable;
import jftplevelrefactoring.looplevel.ui.Progressbar;

/**
 * 1.1.1. Extract Subclass: FTPTransfer extends FTPCmdServer and relevant
 * methods are pushed down. 1.1.2. FTPTransfer extends FTPConnection. 1.5.1.
 * Replace Delegation with Inheritance: FTPTransfer extends FTPDirectory.
 */
public class FTPTransfer extends FTPDirectory implements Transferable {

	/**
	 * 1.1.2. FTPTransfer constructors.
	 */
	public FTPTransfer() {
		super();
	}

	public FTPTransfer(InetAddress server, int port) {
		super(server, port);
	}

	public FTPTransfer(File currentWorkingDir) {
		super(currentWorkingDir);
	}

	public FTPTransfer(InetAddress server, int port, File
currentWorkingDir) {
		super(server, port, currentWorkingDir);
	}

	/**
	 * 1.1.3. Relevant methods are pushed down from FTPCmdServer.
	 */

	/* <START Implementing "Transferable"> */

	/**
	 * Reads data from the FTPServer.
	 *
	 * @return String
	 *@since 0.0.2pA4
	 *@version 3
	 */
	public String readCtrl() {
		String rc = "", line;
		int i = 0;

		try {
			do {
				line = "";

				do {
					i = getCtrlStream().read();
					line += (char) i;
				} while (i != 10);

				rc += line;

			} while ((char) line.charAt(3) != ' ');
		} catch (Exception e) {
			printErrorLine("Error while reading from server (in
fcs): "
```

```
                                                                + e.toString());
            }

            return (rc);
      }

      /**
       * Reads a whole block of data from the ctrl-channel of the FTPServer.
       *
       * @return String
       *@since 0.0.2pA4
       *@version 2
       */
      public String readCtrlBlock() {
            String rc = "";
            int i = 0; // char read from server.

            try {
                  do {
                        i = getCtrlStream().read();
                        if (i == -1)
                              break;
                        rc += (char) i;
                  } while (true);
            } catch (IOException e) {
                  printErrorLine("Error receiving Block.");
            }

            return (rc);
      }

      /**
       * Reads data from the FTPServer.
       *
       * @return String
       *@since 0.0.2pA4
       *@version 1
       */
      public String readData() {
            String rc = "";
            int i = 0; // char read from server.

            try {
                  i = getDataStream().read();

                  while (i != -1) {
                        rc += (char) i;
                        i = getDataStream().read();
                        // printErrorLine("retrieving: " + (char) i);
                  }
                  closePassiveDataConnection();
            } catch (IOException e) {
                  printErrorLine("Error receiving Block.");
            }

            return (rc);
      }

      /**
       * Reads a whole block of data from the data-channel of the FTPServer.
       *
       * @return String
       *@since 0.0.2pA4
```

```
         *@version 0
         */
        public String readDataBlock() {
                String rc = "";

                while ("this".equals("that")) {
                        try {
                                rc += getDataStream().read();
                        } catch (Exception e) {
                                printErrorLine("Gotcha!");
                        }
                } // YES, this does NOT work.

                return (rc);
        }

        /**
         * Reads a single line of data from the data-channel of the FTPServer.
         *
         * @return String
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataLine() {
                String rc = "";

                while ("this".equals("that")) {
                        try {
                                rc += getDataStream().read();
                        } catch (Exception e) {
                                printErrorLine("Gotcha!");
                        }
                } // YES, this does NOT work.

                return (rc);
        }

        /**
         * Sends a String to the FTPServer.
         *
         * @param s
         *             String to send
         *@since 0.0.2pA1
         */
        public void send(String s) {
                getCtrlWriter().write(s);
        }

        /**
         * Receives a list of files named 'localList' and tx's them to the
server.
         *
         * @param localList
         *            a java.lang.String array where to save the files
         *@return true if successfully transfered; else false
         *@since 0.1.99t2
         *@version 1
         */
        public boolean putFiles(String[] localList) {
                boolean rc = true;
                int i;

                for (i = 0; i < localList.length; i++) {
```

```
                    if (!putFile(localList[i]))
                          rc = false;
             }

             return (rc);
      }

      /**
       * Receives a list of files named 'serverList' and stores them in
       * 'localList'
       *
       * @param serverList
       *            a java.lang.String array of files to get
       *@param localList
       *            a java.lang.String array where to save the files
       *@return true if successfully transfered; else false
       *@since 0.1.99t2
       *@version 1
       */
      public boolean getFiles(String[] serverList, String[] localList) {
             boolean rc = true;
             int i;

             if (localList.length == serverList.length)
                    for (i = 0; i < localList.length; i++) {
                          if (!getFile(serverList[i], localList[i]))
                                rc = false;
                    }
             else
                    rc = false;

             return (rc);
      }

      /**
       * 6.1. Split Loop: loop in getFile method cannot be split to two
separate
       * loops because progress bar setting monitors status of writing read
bytes
       * process.
       */
      /**
       * Recieves a File named 'ServerFile' and stores it at 'LocalFile'
       *
       * @param serverFile
       *            The Filename at the Server's side
       *@param localFile
       *            The Filename where it is saved (local)
       *@return True if the file is successfully transfered; else false
       *@since 0.0.2pA1
       *@version 2
       */
      public boolean getFile(String serverFile, String localFile) {
             // We must catch errors like '5xx Permission denied...'
             boolean rc = false;
             int size = -1, i = 0;
             String serverResponse = "";
             FileOutputStream fos = null;

             size = getFileSize(serverFile);
             // printOutputLine("Filesize is " + size);// Should only be
used for
             // debug(tk)
```

```
            if (openPassiveDataConnection()) {
                  if (size > 0) {
                        /*
                         * Here we may look first if the file already
exists on the
                         * local fs, compare the size and may only
transfer the missing
                         * part(s).(tk)
                         */
                        send("retr " + serverFile);
                        // printOutputLine( "retrieving file" );//debug

                        // Checking return-value
                        serverResponse = readCtrl();
                        printOutputLine(serverResponse);

                        if (serverResponse.startsWith("550")) {
                              printOutputLine("Permission denied.");
                        } else if (fileStatusIsOkay(serverResponse)) { //
Server is
                              // ready to
                              // transfer.
                              try {
                                    Progressbar      pg      =      new
Progressbar(size);
                                    DataInputStream     dis     =     new
DataInputStream(
                                                getDataInputStream());
                                    i = 0;

                                    fos               =               new
FileOutputStream(getCurrentWorkingDir()
                                                + localFile);

                                    /*
                                     * This should be improved because
single byte reading
                                     *  and  writing  is  _VERY_  slow.
Maybe, there is a kind of
                                     *    blockread    and    blockwrite
somewhere or we can connect
                                     * directly the dis and fos...?
                                     */
                                    /*    FML:    Whats    about    using
BufferedReader/Writer, too */
                                    /**
                                     * 6.1. One loop is used to write
read bytes and show
                                     * progress bar of writing process.
                                     */
                                    while (i < size) {
                                          fos.write((byte) dis.read());
                                          i++;
                                          if ((i % 1024) == 0) // 'hope
this will save some
                                                // CPU-cycles
                                                pg.set(i);
                                    }
                                    pg.set(i); // set the Value again to
get 100% even on
                                    // small files.
```

```
                                        closePassiveDataConnection();
                                        fos.close();
                                        dis.close();
                                        rc = true;
                                  } catch (Exception e) {
                                        printErrorLine("Error  while  Saving
File:"
                                                    + e.toString());
                                  }
                            }

                            // Closing socket && receiving server reply
                            try {
                                  printOutputLine(readCtrl());
                                  if (isPassiveConnected())
                                        closePassiveDataConnection();
                            } catch (Exception e) {
                                  printErrorLine("Can't   Read   from   ctrl-
channel.");
                            }
                      }
                }

                return (rc);
          }

          /**
           * 6.1. Split Loop: loop in putFile method cannot be split to two
separate
           * loops because progress bar setting monitors status of writing read
bytes
           * process.
           */
          /**
           * Puts a File 'localFile' to the Server
           *
           * @param localFile
           *            The file to transmit
           *@return  true  if  the  file  was  transfered  successfully;  otherwise
false
           *@since 0.0.2pA1
           *@version 5
           */
          public boolean putFile(String localFile) {
                // We must catch errors like '5xx Permission denied...'
                DataOutputStream dos;
                FileInputStream fis;
                int size, i;

                if (!openPassiveDataConnection()) {
                      printOutputLine("Could    not    Open    Passive    Data
connection");
                      return (false);
                }
                send("stor " + localFile);
                printOutputLine(readCtrl());

                try {
                      dos = new DataOutputStream(getDataOutputStream());
                      i = 0;
                      fis   =   new   FileInputStream(getCurrentWorkingDir()   +
localFile);
                      /*
```

```
                     * This also should be improved because single byte
writing is
                     * _VERY_ slow.
                     */
                    size = fis.available();

                    Progressbar pb = new Progressbar(size);

                    /**
                     * 6.1. One loop is used to write read bytes and show
progress bar
                     * of writing process.
                     */
                    while (i < size) {
                           dos.write((byte) fis.read());
                           i++;
                           if ((i % 1024) == 0)
                                  pb.set(i);
                    }
                    pb.set(i);

                    closePassiveDataConnection();
                    fis.close();
                    dos.close();
                    // FML: What about reading "270 ..." ?
                    printOutputLine(readCtrl());
             } catch (Exception e) {
                    printOutputLine("Exception occured while sending data to
the server: "
                                  + e.toString());
             }

             /* closePassiveDataConnection(); // FML: How often you want to
do this? */

             /* return ( false ); // FML: SURE???? */
             return (true);
       }

       /* <START Implementing tools for "Transferable"> */

       /* May someone improve this ugly code ? PLEASE */
       /**
        * Returns the size of a file named 'file'. It first tries to use the
size
        * command and falls back to a selfdetection mode by parsing the
output of a
        * 'list' command. If even this is not supported by the server it
fails.
        *
        * @param file
        *            The name of the file
        *@return The size of the given file; or -1 if it can't detect it.
        *@since 0.1.99
        *@version 3
        */
       private int getFileSize(String file) {
              /*
               * At first I'll try the size command, if it fails the method
tries to
               * determinate(?) the filesize by parsing the output of 'list'.
               */
              int rc = -1;
```

```
		String receive;

		send("size " + file);
		receive = readCtrl();

		if (!fileActionIsNotTaken(receive)) {
			try {
				if (fileStatusIsReplied(receive))
					receive = receive.substring(3).trim();

				rc = Integer.parseInt(receive);
			} catch (Exception e) {
				printErrorLine("Server responses an invalid
filesize.");
			}
		} else // Now me must go the hard way. :-(
		{
			int i;
			String list;

			list = getLongDirList();

			// Cutting the 1st and last line ("150 .." && "226 ..")
			list = list.substring(list.indexOf("\n") + 1, // after
1st NL
					list.lastIndexOf("\n", list.length() - 2)//
before last NL
					);

			/*
			 * Now we should have cut the first and last line. At
next we have
			 * to to replace '\n''s by ' ', so that the tokenizer
works proper.
			 * ARRG!, Then we have to replace '\r''s with ' ''s,
too.
			 */
			list = list.replace('\n', ' ');
			list = list.replace('\r', ' '); // Yes..

			// Now we've got the String we want...going on.
			StringTokenizer st = new StringTokenizer(list, " ");
			String token;

			boolean firstRun = true;
			int size = 0;
			int j = 1;

			/*
			 * When tokenizing the first line, the needed fields are
8 & 4 and
			 * in any further run they are 9 & 5. Don't ask me
why... but tell
			 * me if you know it! ;-)
			 */
			for (i = 0; st.hasMoreTokens(); i++) {
				token = st.nextToken();

				if (i == (9 - j)) {
					i = 0;
					if (firstRun) {
						firstRun = false;
						j = 0;
```

```
                                }
                                if (token.equals(file)) {
                                        //  Since  '(5-j)'  is  smaller  than
'(9-j)' we can assign
                                        // 'size' here.
                                        rc = size;
                                        break;
                                }
                        }
                        if (i == (5 - j))
                                size = Integer.parseInt(token);
                }
        }

        return (rc);
    }

    /**
     * Returns the current InputStreamReader of the ctrl-channel.
     *
     * @return InputStreamReader from the current ctrl-channel
     *@since 0.0.2pA4
     *@version 0
     */
    private InputStreamReader getCtrlStream() {
        if (getCtrlIsr() == null) {
            this.setCtrlIsr(getCtrlReader().getStream());
        }

        return (getCtrlIsr());
    }

    /**
     * Returns the current InputStreamReader of the data-channel.
     *
     * @return InputStreamReader from the current data-channel
     *@since 0.1.0
     *@version 0
     */
    private InputStreamReader getDataStream() {
        if (getDataIsr() == null) {
            this.setDataIsr(getDataReader().getStream());
        }

        return (getDataIsr());
    }

    /**
     * Returns the current InputStream of the data-channel.
     *
     * @return InputStream from the current data-channel
     *@since 0.1.99
     *@version 0
     */
    private InputStream getDataInputStream() {
        if (getDataIs() == null) {
            this.setDataIs(getDataReader().getInputStream());
        }

        return (getDataIs());
    }

    /**
```

```
         * Returns the current OutputStream of the data-channel.
         *
         * @return OutputStream from the current data-channel
         *@since 0.1.99
         *@version 0
         */
        private OutputStream getDataOutputStream() {
                if (getDataOs() == null) {
                        this.setDataOs(getDataWriter().getOutputStream());
                }

                return (getDataOs());
        }

        /* <END Implementing tools for "Transferable"> */
        /* <END Implementing "Transferable"> */

        /**
         *  1.4.  Hide  Delegate:  FTPDirectory  is  hidden  by  getLongDirList
method.
         * 1.5.2. getLongDirList method is removed due to replacing delegation
to
         * inheritance.
         */
        /*
         * protected String getLongDirList() { // File currentDirectory = new
         *   File(this.getCurrentWorkingDir());   FTPDirectory   dir   =   new
FTPDirectory();
         * return dir.getLongDirList(); }
         */

        /**
         * 4.1. Extract Method: printOutputLine and printErrorLine methods.
         */
        /**
         * 4.4. Pull Up Method: printOutputLine and printErrorLine methods are
         * pulled up to FTPConnection.
         */
        /*
         * protected void printOutputLine(String message) {
         * getStdout().println(message); }
         *
         * protected void printErrorLine(String message) {
         * getStderr().println(message); }
         */

        /**
         * 4.2. Decompose Conditional: fileStatusIsReplied method.
         */
        private boolean fileStatusIsReplied(String response) {
                return  (response.startsWith("213")  &&  response.charAt(3)  ==  '
');
        }

        /**
         * 4.2. Decompose Conditional: fileStatusIsOkay method.
         */
        private boolean fileStatusIsOkay(String response) {
                return (response.startsWith("150"));
        }
}

/*
```

```
 * Name:    JFTPSuper.java
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name               changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz       Creation
 * 0.0.2pA2 30.04.2001 Tobias Kranz       added methods to handle variables/
 *                                        options
 *   0.0.2pA4   15.05.2001   Tobias   Kranz                added   option   for
'PassiveConnected'
 *   0.0.2pA4   28.05.2001   Tobias   Kranz               added   debuginfo   for
'setServerTo(..)'
 * 0.0.2pA4 05.06.2001 Tobias Kranz       added 'getLicense()' &
 *                                        'getCopyright()' method's
 * 0.0.2pA4 06.06.2001 Tobias Kranz       removed debuginfo for
 *                                        'setServerTo(..)'
 * 0.0.2pA5 07.06.2001 Sebastian Schipper fixed serious bug in status-
 *                                        methods
 * 0.1.99t4 20.07.2001 Tobias Kranz       added 'getCurrentWorkingDir()' &&
 *                                        'setCurrentWorkingDir(String)' &&
 *                                        'String getDirSeparator()'
 * 0.1.99t5 23.07.2001 Tobias Kranz       fixed 'setCurrentWorkingDir()' &&
 *                                        removed 'getDirSeparator()' (see
 *                                        JavaDoc:java.io.File for details)
 */
/**
 * 2.1. Extract Package: core package.
 */
package jftplevelrefactoring.looplevel.core;

/*import java.io.File;
 import java.net.InetAddress;
 import java.net.UnknownHostException;
 import java.util.BitSet; import java.util.StringTokenizer;

 import jftplevelrefactoring.looplevel.io.StdErr;*/
import jftplevelrefactoring.looplevel.io.StdOut;

/**
 * Parameter & Object class.
 *
 * @since 0.0.2pA1
 */
public class JFTPSuper {
       /**
        * 1.1.3.1. Remove static variable jftpSuper and remove inheritance
        * relationships to JFTPSuper which only rely on jftpSuper.
        */
       /**/
       // protected static JFTPSuper jftpSuper = new JFTPSuper();
       /**/

       private final String buildDate = "01.08.2001";
       private final String version = "0.1.99test6";

       /**
        * 3.3.1. Push Down Field: connected and passiveConnected options are
pushed
        * down with encapsulation methods to FTPConnection.
        */
       // private BitSet options = new BitSet(4);
       /**
        * 3.3.1. Push Down Field: server and port are pushed down with
```

```
         * encapsulation methods to FTPConnection.
         */
        // private InetAddress server;
        // private int port = 21;
        /**
         * 3.3.1. Push Down Field: userName, anonymousPasswd and authenticated
are
         * pushed down with encapsulation methods to FTPAuthentication.
         */
        // private String userName = "me";
        // private String anonymousPasswd = "tux@northpole.org";
        // private boolean authenticated = false;
        /**
         * 3.3.1. Push Down Field: currentWorkingDir is pushed down with
         * encapsulation methods to FTPDirectory and FTPTransfer.
         */
        // private File currentWorkingDir = new File(".");
        private short debugLevel = 0;
        private String info = "JavaFTP-Client v. " + version
                        + "                                                Build: " +
buildDate;

        public JFTPSuper() {
                /*
                 * BitSet 'options'
                 *
                 * #: if true: default: 0 => cli true, 1 => passive true, 2 =>
anonymous
                 * false, 3 => autoconnect false.
                 */
                /*
                 *           this.options.set(0);           this.options.set(1);
this.options.clear(2);
                 * this.options.clear(3);
                 */

                /*
                 *  try  {  this.server  =  InetAddress.getByName("localhost");  }
catch
                 *                   (UnknownHostException                   uhe)
{ System.err.println("COUGHT!"); }
                 */
        }

        /*     *//**
         * Is Cli-mode running ?
         *
         * @return true if Cli-mode is running; false if not
         *@since 0.0.2pA1
         */
        /*
         * public boolean isCli() { return ((boolean) options.get(0)); }
         *//**
         * Sets Cli-mode to true.
         *
         * @since 0.0.2pA1
         */
        /*
         * public void setCli() { options.set(0); }
         *//**
         * Sets Cli-mode to false.
         *
         * @since 0.0.2pA1
```

```
         */
        /*
         * protected void unsetCli() { options.clear(0); }
         *//**
         * Is Gui-mode running ?
         *
         * @return true if isCli() returns false and other way round.
         *@since 0.0.2pA1
         */
        /*
         *  public  boolean  isGui()  {  return  (((boolean)  options.get(0)  ?
(false) :
         * (true))); }
         *//**
         * Same than unsetCli.
         *
         * @since 0.0.2pA1
         */
        /*
         * public void setGui() { unsetCli(); }
         *//**
         * Is passive connection mode prefered ?
         *
         * @return true if passive mode is prefered; false if not
         *@since 0.0.2pA1
         */
        /*
         *    protected    boolean    isPassive()    {    return    ((boolean)
options.get(1)); }
         *//**
         * Sets passive mode to true.
         *
         * @since 0.0.2pA1
         */
        /*
         * public void setPassive() { options.set(1); }
         *//**
         * Sets passive mode to false.
         *
         * @since 0.0.2pA1
         */
        /*
         * protected void unsetPassive() { options.clear(1); }
         *//**
         * Is anonymous logon prefered ?
         *
         * @return true if anonymous logon is prefered; otherwise false
         *@since 0.0.2pA1
         */
        /*
         * public boolean isAnonymous() { return ((boolean) options.get(2)); }
         *//**
         * Sets anonymous logon to true.
         *
         * @since 0.0.2pA1
         */
        /*
         * public void setAnonymous() { options.set(2); }
         *//**
         * Sets anonymous logon to false.
         *
         * @since 0.0.2pA1
         */
```

```
       /*
        * protected void unsetAnonymous() { options.clear(2); }
        *//**
        * Returns Autoconnect
        *
        * @return true if Autoconnect is true; otherwise false
        *@since 0.0.2pA2
        */
       /*
        *    public    boolean    isAutoconnect()    {    return    ((boolean)
options.get(3)); }
        *//**
        * Sets Autoconnect to true.
        *
        * @since 0.0.2pA2
        */
       /*
        * public void setAutoconnect() { options.set(3); }
        *//**
        * Sets Autoconnect to false.
        *
        * @since 0.0.2pA2
        */
       /*
        * protected void unsetAutoconnect() { options.clear(3); }
        */

       /*     *//**
        * Returns the current Server
        *
        * @return InetAddress of the Server
        *@since 0.0.2pA1
        */
       /*
        * protected InetAddress getServer() { return (server); }
        *//**
        * Returns the IP of the Server
        *
        * @return the servers IP
        *@since 0.0.2pA1
        */
       /*
        * public String getServerIP() { return (server.getHostAddress()); }
        *//**
        * Returns the Name of the Server
        *
        * @return the servers Name
        *@since 0.0.2pA1
        */
       /*
        * protected String getServerName() { return (server.getHostName()); }
        *//**
        * sets the Servers IP/Name
        *
        * @param s
        *             Servers name/IP
        *@return true if the Server can be set; otherwise false
        *@version 2
        *@since 0.0.2pA1
        */
       /*
        * public boolean setServerTo(String s) { boolean rc;
        *
```

```
         * try { server = InetAddress.getByName(s);
         *
         * rc = true; } catch (UnknownHostException uhe) { StdErr stderr = new
         * StdErr(); stderr.println("Unable to reach host '" + s + "'."); rc =
         * false; }
         *
         * return (rc); }
         *//**
         * What's the current port?
         *
         * @return The current port
         *@since 0.0.2pA1
         */
        /*
         * public int getServerPort() { return (port); }
         *//**
         * Sets the port
         *
         * @param Port
         *            Port
         *@since 0.0.2pA1
         */
        /*
         * public void setServerPortTo(int Port) { port = Port; }
         *//**
         * Returns the userName
         *
         * @return The current userName
         *@since 0.0.2pA2
         */
        /*
         * protected String getUserName() { return (userName); }
         *//**
         * Sets the userName
         *
         * @param s
         *            New userName
         *@since 0.0.2pA2
         */
        /*
         * protected void setUserName(String s) { userName = s; }
         *//**
         * Returns the anonymous passwd.
         *
         * @return The password used for anonymous login.
         *@since 0.0.2pA2
         */
        /*
         * public String getAnonymousPasswd() { return (anonymousPasswd); }
         *//**
         * Sets the password used for anonymous login to s.
         *
         * @param s
         *            The password used for anonymous login.
         *@since 0.0.2pA2
         */
        /*
         * public void setAnonymousPasswdTo(String s) { anonymousPasswd = s; }
         *//**
         * Are we authenticated ?
         *
         * @return true if we are authenticated; else false
         *@since 0.0.2pA2
```

```
         */
        /*
         * protected boolean isAuthenticated() { return (authenticated); }
         *//**
         * Sets authenticated to true
         *
         * @since 0.0.2pA2
         */
        /*
         * protected void setAuthenticated() { authenticated = true; }
         *//**
         * Sets authenticated to false
         *
         * @since 0.0.2pA2
         */
        /*
         * protected void unsetAuthenticated() { authenticated = false; }
         *//**
         * Returns a java.lang.String containing the current working directory
         *
         * @return A java.lang.String
         *@since 0.1.99t4
         *@version 1
         */
        /*
         * protected String getCurrentWorkingDir() { String rc =
         *    currentWorkingDir.getAbsolutePath();    rc    =    rc.substring(0,
rc.length() -
         * 1);
         *
         * return (rc); }
         *//**
         * Sets the current working dir to the given java.lang.String
         *
         * @param A
         *            java.lang.String
         *@return true if path is set; otherwise false
         *@since 0.1.99t4
         *@version 2
         */
        /*
         *
         * Some infos: Java's path-format is
         *  'DIR_SEPARATOR''PATH''DIR_SEPARATOR''.'.  eg:  /  PATH  /  .  .  eg:
"/root/."
         * || "C:\MICROS~1\."
         *
         * protected boolean setCurrentWorkingDir(String newPath) { boolean rc
=
         * true; String cp;
         *
         * if (newPath.startsWith("/")) // absolute { if (!(new
         * File(newPath)).exists()) rc = false; else currentWorkingDir = new
         * File(newPath + File.separator + "."); } else if
         * (newPath.startsWith("..")) // Dir up { cp = getCurrentWorkingDir();
int
         *  lastIndex  =  cp.length()  -  2;  //  cutting  '\n'  from  the  end
StringTokenizer
         * st = new StringTokenizer(newPath, File.separator);
         *
         *   do   {   lastIndex   =   cp.lastIndexOf("/",   lastIndex);   cp   =
cp.substring(0,
         * lastIndex); // pop ".." to /dev/null (we just need the 'st' as
```

```
counter)
        * st.nextToken(); } while ((st.hasMoreTokens()));
        *
        * cp += File.separator + ".";
        *
        * if (!(new File(cp)).exists()) rc = false; else currentWorkingDir =
new
        * File(cp); } else // relative { cp = getCurrentWorkingDir(); if
        * (cp.endsWith(".")) cp = cp.substring(0, cp.length() - 1);
        *
        * if (!(new File(cp + newPath + File.separator + ".")).exists()) rc =
        *  false;  else  currentWorkingDir  =  new  File(cp  +  newPath  +
File.separator +
        * "."); }
        *
        * return (rc); }
        */

       /**
        * Returns the debugLevel
        *
        * @return The current debugLevel
        *@since 0.0.2pA1
        */
       public short getDebugLevel() {
              return ((short) debugLevel);
       }

       /**
        * Sets the debugLevel to the short-value of 's'
        *
        * @param s
        *            New debugLevel
        *@since 0.0.2pA1
        */
       public void setDebugLevel(short s) {
              debugLevel = s;
       }

       /**
        * Returns a 80 character long info string.
        *
        * @return Info 'bout JFTP
        */
       public String getJFTPInfo() {
              return (info);
       }

       /**
        * Returns a very long info string.
        *
        * @return Info 'bout JFTP
        */
       protected String getLongJFTPInfo() {
              return ("\n"
                            + info
                            + "\n\n(c) 2000 - 2001 The JavaFTP-Group:\n\n"
                            + "    Uwe         Busch\n"
                            + "    Alexander   Flick\n"
                            + "    Christian   Kaminski\n"
                            + "    Tobias      Kranz       (TOBx@GMX.DE)\n"
                            + "    Martin      Loh         (MLoh80@GMX.DE)\n"
                            + "    Sebastian   Schipper    (dpi209@GMX.DE)\n"
```

```
                            + "    Mathias     Sroke\n"
                            +   "              Dolapo                    Falola
(Dr_Steelface@HOTMAIL.COM)\n"
                            + "    Svenja      Wittstadt\n"
                            + "\n"
                            + "This software is distributed under the rules of
the General Public " + "License(GPL) version 2 or any later version.\n");
      }

      /**
       * Returns the License valid for this application
       *
       * @return A String containing the used License
       *@since 0.0.2pA4
       *@version 1
       */
      public String getLicense() {
            return (" GNU GENERAL PUBLIC LICENSE\n Version 2, June 1991\n
\n Copyright (C) 1989, 1991 Free Software Foundation, Inc.\n 675 Mass Ave,
Cambridge, MA 02139, USA\n Everyone is permitted to copy and distribute
verbatim copies\n of this license document, but changing it is not
allowed.\n \n Preamble\n \n The licenses for most software are designed to
take away your\n freedom to share and change it.  By contrast, the GNU
General Public\n License is intended to guarantee your freedom to share and
change free\n software--to make sure the software is free for all its users.
This\n General Public License applies to most of the Free Software\n
Foundation's software and to any other program whose authors commit to\n
using it.  (Some other Free Software Foundation software is covered by\n the
GNU Library General Public License instead.)  You can apply it to\n your
programs, too.\n \n When we speak of free software, we are referring to
freedom, not\n price.  Our General Public Licenses are designed to make sure
that you\n have the freedom to distribute copies of free software (and
charge for\n this service if you wish), that you receive source code or can
get it\n if you want it, that you can change the software or use pieces of
it\n in new free programs; and that you know you can do these things.\n \n
To protect your rights, we need to make restrictions that forbid\n anyone to
deny you these rights or to ask you to surrender the rights.\n These
restrictions translate to certain responsibilities for you if you\n
distribute copies of the software, or if you modify it.\n \n For example, if
you distribute copies of such a program, whether\n gratis or for a fee, you
must give the recipients all the rights that\n you have.  You must make sure
that they, too, receive or can get the\n source code.  And you must show
them these terms so they know their\n rights.\n \n We protect your rights
with two steps: (1) copyright the software, and\n (2) offer you this license
which gives you legal permission to copy,\n distribute and/or modify the
software.\n \n Also, for each author's protection and ours, we want to make
certain\n that everyone understands that there is no warranty for this
free\n software.  If the software is modified by someone else and passed on,
we\n want its recipients to know that what they have is not the original,
so\n that any problems introduced by others will not reflect on the
original\n authors' reputations.\n \n Finally, any free program is
threatened constantly by software\n patents.  We wish to avoid the danger
that redistributors of a free\n program will individually obtain patent
licenses, in effect making the\n program proprietary.  To prevent this, we
have made it clear that any\n patent must be licensed for everyone's free
use or not licensed at all.\n \n The precise terms and conditions for
copying, distribution and\n modification follow.\n \n GNU GENERAL PUBLIC
LICENSE\n TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION\n
\n 0. This License applies to any program or other work which contains\n a
notice placed by the copyright holder saying it may be distributed\n under
the terms of this General Public License.  The \"Program\", below,\n refers
to any such program or work, and a \"work based on the Program\"\n means
either the Program or any derivative work under copyright law:\n that is to
```

```
say, a work containing the Program or a portion of it,\n either verbatim or with modifications and/or translated into another\n language.  (Hereinafter, translation is included without limitation in\n the term \"modification\".) Each licensee is addressed as \"you\".\n \n Activities other than copying, distribution and modification are not\n covered by this License; they are outside its scope.  The act of\n running the Program is not restricted, and the output from the Program\n is covered only if its contents constitute a work based on the\n Program (independent of having been made by running the Program).\n Whether that is true depends on what the Program does.\n \n 1. You may copy and distribute verbatim copies of the Program's\n source code as you receive it, in any medium, provided that you\n conspicuously and appropriately publish on each copy an appropriate\n copyright notice and disclaimer of warranty; keep intact all the\n notices that refer to this License and to the absence of any warranty;\n and give any other recipients of the Program a copy of this License\n along with the Program.\n \n You may charge a fee for the physical act of transferring a copy, and\n you may at your option offer warranty protection in exchange for a fee.\n \n 2. You may modify your copy or copies of the Program or any portion\n of it, thus forming a work based on the Program, and copy and\n distribute such modifications or work under the terms of Section 1\n above, provided that you also meet all of these conditions:\n \n a) You must cause the modified files to carry prominent notices\n stating that you changed the files and the date of any change.\n \n b) You must cause any work that you distribute or publish, that in\n \nwhole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third\n parties under the terms of this License.\n \n c) If the modified program normally reads commands interactively\n when run, you must cause it, when started running for such\n interactive use in the most ordinary way, to print or display an\n announcement including an appropriate copyright notice and a\n notice that there is no warranty (or else, saying that you provide\n a warranty) and that users may redistribute the program under\n these conditions, and telling the user how to view a copy of this\n License. (Exception: if the Program itself is interactive but\n does not normally print such an announcement, your work based on\n the Program is not required to print an announcement.)\n \n These requirements apply to the modified work as a whole.  If\n identifiable sections of that work are not derived from the Program,\n and can be reasonably considered independent and separate works in\n themselves, then this License, and its terms, do not apply to those\n sections when you distribute them as separate works.  But when you\n distribute the same sections as part of a whole which is a work based\n on the Program, the distribution of the whole must be on the terms of\n this License, whose permissions for other licensees extend to the\n entire whole, and thus to each and every part regardless of who wrote it.\n \n Thus, it is not the intent of this section to claim rights or contest\n your rights to work written entirely by you; rather, the intent is to\n exercise the right to control the distribution of derivative or\n collective works based on the Program.\n \n In addition, mere aggregation of another work not based on the Program\n with the Program (or with a work based on the Program) on a volume of\n a storage or distribution medium does not bring the other work under\n the scope of this License.\n \n 3. You may copy and distribute the Program (or a work based on it,\n under Section 2) in object code or executable form under the terms of\n Sections 1 and 2 above provided that you also do one of the following:\n \n a) Accompany it with the complete corresponding machine-readable\n source code, which must be distributed under the terms of Sections\n 1 and 2 above on a medium customarily used for software interchange; or,\n \n b) Accompany it with a written offer, valid for at least three\n years, to give any third party, for a charge no more than your\n cost of physically performing source distribution, a complete\n machine-readable copy of the corresponding source code, to be\n distributed under the terms of Sections 1 and 2 above on a medium\n customarily used for software interchange; or,\n \n c) Accompany it with the information you received as to the offer\n to distribute corresponding source code.  (This alternative is\n allowed only for
```

noncommercial distribution and only if you\n received the program in object code or executable form with such\n an offer, in accord with Subsection b above.)\n \n The source code for a work means the preferred form of the work for\n making modifications to it. For an executable work, complete source\n code means all the source code for all modules it contains, plus any\n associated interface definition files, plus the scripts used to\n control compilation and installation of the executable. However, as a\n special exception, the source code distributed need not include\n anything that is normally distributed (in either source or binary\n form) with the major components (compiler, kernel, and so on) of the\n operating system on which the executable runs, unless that component\n itself accompanies the executable.\n \n If distribution of executable or object code is made by offering\n access to copy from a designated place, then offering equivalent\n access to copy the source code from the same place counts as\n distribution of the source code, even though third parties are not\n compelled to copy the source along with the object code.\n \n 4. You may not copy, modify, sublicense, or distribute the Program\n except as expressly provided under this License. Any attempt\n otherwise to copy, modify, sublicense or distribute the Program is\n void, and will automatically terminate your rights under this License.\n However, parties who have received copies, or rights, from you under\n this License will not have their licenses terminated so long as such\n parties remain in full compliance.\n \n 5. You are not required to accept this License, since you have not\n signed it. However, nothing else grants you permission to modify or\n distribute the Program or its derivative works. These actions are\n prohibited by law if you do not accept this License. Therefore, by\n modifying or distributing the Program (or any work based on the\n Program), you indicate your acceptance of this License to do so, and\n all its terms and conditions for copying, distributing or modifying\n the Program or works based on it.\n \n 6. Each time you redistribute the Program (or any work based on the\n Program), the recipient automatically receives a license from the\n original licensor to copy, distribute or modify the Program subject to\n these terms and conditions. You may not impose any further\n restrictions on the recipients' exercise of the rights granted herein.\n You are not responsible for enforcing compliance by third parties to\n \nthis License. \n \n7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues),\n conditions are imposed on you (whether by court order, agreement or\n otherwise) that contradict the conditions of this License, they do not\n excuse you from the conditions of this License. If you cannot\n distribute so as to satisfy simultaneously your obligations under this\n License and any other pertinent obligations, then as a consequence you\n may not distribute the Program at all. For example, if a patent\n license would not permit royalty-free redistribution of the Program by\n all those who receive copies directly or indirectly through you, then\n the only way you could satisfy both it and this License would be to\n refrain entirely from distribution of the Program.\n \n If any portion of this section is held invalid or unenforceable under\n any particular circumstance, the balance of the section is intended to\n apply and the section as a whole is intended to apply in other\n circumstances.\n \n It is not the purpose of this section to induce you to infringe any\n patents or other property right claims or to contest validity of any\n such claims; this section has the sole purpose of protecting the\n integrity of the free software distribution system, which is\n implemented by public license practices. Many people have made\n generous contributions to the wide range of software distributed\n through that system in reliance on consistent application of that\n system; it is up to the author/donor to decide if he or she is willing\n to distribute software through any other system and a licensee cannot\n impose that choice.\n \n This section is intended to make thoroughly clear what is believed to\n be a consequence of the rest of this License.\n \n 8. If the distribution and/or use of the Program is restricted in\n certain countries either by patents or by copyrighted interfaces, the\n original copyright holder who places the Program under this License\n may add an explicit

```
geographical distribution limitation excluding\n those countries, so that distribution is permitted only in or among\n countries not thus excluded. In such case, this License incorporates\n the limitation as if written in the body of this License.\n \n 9. The Free Software Foundation may publish revised and/or new versions\n of the General Public License from time to time.  Such new versions will\n be similar in spirit to the present version, but may differ in detail to\n address new problems or concerns.\n \n Each version is given a distinguishing version number.  If the Program\n specifies a version number of this License which applies to it and \"any\n later version\", you have the option of following the terms and conditions\n either of that version or of any later version published by the Free\n Software Foundation.  If the Program does not specify a version number of\n this License, you may choose any version ever published by the Free Software\n Foundation.\n \n 10. If you wish to incorporate parts of the Program into other free\n programs whose distribution conditions are different, write to the author\n to ask for permission.  For software which is copyrighted by the Free\n Software Foundation, write to the Free Software Foundation; we sometimes\n make exceptions for this.  Our decision will be guided by the two goals\n of preserving the free status of all derivatives of our free software and\n of promoting the sharing and reuse of software generally.\n \n NO WARRANTY\n \n 11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY\n FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN\n OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES\n PROVIDE THE PROGRAM \"AS IS\" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED\n OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF\n MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS\n TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE\n PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,\n REPAIR OR CORRECTION.\n \n 12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING\n WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR\n REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,\n INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING\n OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED\n TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY\n YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER\n PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE\n POSSIBILITY OF SUCH DAMAGES.\n \n END OF TERMS AND CONDITIONS");
	}

	/**
	 * Returns a short information about the License
	 *
	 * @return A String containing a short information about the used
license
	 *@since 0.0.2pA4
	 *@version 0
	 */
	public String getLicenseInfo() {
		return ("This Software comes without any warranty.\n"
				+ "You may redistribute and/or modify it under the
terms of the GPL v 2.\n"
				+ "For details type 'license'.\n");
	}

	/**
	 * Returns the copyright informations valid for this software
	 *
	 * @return A String containing copyright informations
	 *@since 0.0.2pA4
	 *@version 0
	 */
	public String getCopyright() {
```

```
              return ("Copyright 2000 - 2001 The JavaFTP-Group.\n"
                           + "For details type 'info'.\n");
       }

       /**
        * Easter egg
        *
        * @since 0.0.2pA4
        *@version [.,xX@Xx,.]
        */
       public void printLongJFTPInfo() {
              final boolean z = false;
              int i = ((!!!!!z) ? (-1) : (0));
              try {
                     for (;;) {
                           Thread.sleep(((!!!!!z) ? (0) : (1)), ((!!!!z) ?
(0) : (1)));
                           (new
StdOut()).print(getLongJFTPInfo().charAt(++i));
                           if (i >= getLongJFTPInfo().length())
                                  break;
                     }
              } catch (Exception e) {
              }
       }
}

/*
 * Name:    Transferable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 *  0.0.2pA2  03.05.2001  Tobias  Kranz  Changed  'receive-Line/-Block'  to
'receive()'
 * 0.1.0    09.06.2001 Martin Loh   Changed the 'getFile()' and 'putFile()'
 *                                          method definitions so that they
 *                                                 return an boolean and have
Strings
 *                                          as args.
 * 0.1.99t2 17.07.2001 Tobias Kranz Added 'getFiles(String[], String[])',
 *                                  corrected some comments.
 * 0.1.99t4 19.07.2001 Tobias Kranz cleaned up
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 * Defines methods to transfer "raw" data and files.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface is in-lined into Transferable.
 * 1.6.2. Inheritance relationship is removed because it is not used.
 */
public interface Transferable {
       /**
        * Reads data from the ctrl-channel.
        *
```

```
         * @return The data read.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readCtrl();

        /**
         * Receives a whole block of data.
         *
         * @return The block received
         *@since 0.0.2pA4
         *@version 0
         */
        public String readCtrlBlock();

        /**
         * Reads data until end of Stream
         *
         * @return The data read.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readData();

        /**
         * Reads a whole block of data.
         *
         * @return The block received
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataBlock();

        /**
         * Reads a single line of data.
         *
         * @return The data received.
         *@since 0.0.2pA4
         *@version 0
         */
        public String readDataLine();

        /**
         * Recieves a list of Files 'serverlist' and saves it to 'localList'.
         *
         * @param serverList
         *            Filelist (java.lang.String array)
         *@param localList
         *            Filelist (java.lang.String array)
         *@return true if successfully transfered; otherwise false.
         *@since 0.1.99t2
         *@version 1
         */
        public boolean getFiles(String[] serverList, String[] localList);

        /**
         * Recieves a File with name 'Serverfile' and saves it to 'localfile'.
         *
         * @param ServerFile
         *            Filename
         *@param LocalFile
         *            Filename
         *@return true if successfully transfered; otherwise false.
```

```
        *@since 0.0.2pA1
        *@version 1
        */
       public boolean getFile(String ServerFile, String LocalFile);

       /**
        * Puts a File named 'LocalFile'.
        *
        * @param LocalFile
        *            Filename
        *@return true if the file was transfered successfully; false if not.
        *@since 0.0.2pA1
        *@version 0
        */
       public boolean putFile(String LocalFile);

       /**
        * Sends a string of data.
        *
        * @param s
        *            String to send.
        *@since 0.0.2pA1
        *@version 0
        */
       public void send(String s);
}

/*
 * Name:    Authenticateable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 17.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 06.06.2001 Tobias Kranz Changed name to __REAL__ english ;-)
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 * Defines methods to authenticate a user.
 *
 * @since 0.0.2pA1
 */
/**
 *    1.6.1.    Inline    Class:    JFTPSuperInterface    is    in-lined    into
Authenticateable.
 * 1.6.2. Inheritance relationship is removed because it is not used.
 */
public interface Authenticateable {
       /**
        * Authenticates an user.
        *
        * @return true if authentication succeds; otherwise false
        *@since 0.0.2pA1
        */
       public boolean authenticate();
}

/*
 * Name:    Connectable.java  [INTERFACE] (extends JFTPSuperInterface)
```

```
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 * Defines methods to connect to a server.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface is in-lined into Connectable.
 * 1.6.2. Inheritance relationship is removed because it is not used.
 */
public interface Connectable {
       public boolean connect();

       public boolean disconnect();

       /**
        * Opens a ctrl connection.
        *
        * @since 0.0.2pA4
        */
       public boolean openCtrlConnection();

       /**
        * Closes a ctrl connection.
        *
        * @since 0.0.2pA4
        */
       public boolean closeCtrlConnection();

       /**
        * Opens a passive data connection.
        *
        * @since 0.0.2pA4
        */
       public boolean openPassiveDataConnection();

       /**
        * Closes a passive data connection.
        *
        * @since 0.0.2pA4
        */
       public boolean closePassiveDataConnection();
}

/*
 * Name:    Createable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 19.07.2001 Tobias Kranz Creation
 */
/**
```

```
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 * Defines methods to create directories and files.
 *
 * @since 0.1.99t4
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface is in-lined into Createable.
 * 1.6.2. Inheritance relationship is removed because it is not used.
 */
public interface Createable {
       /**
        * Creates a Directory on the server with the given name
        *
        * @return True if successfully created; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean createRemoteDir(String dirName);

       /**
        * Creates a Directory on the local fs with the given name
        *
        * @return True if successfully created; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean createLocalDir(String dirName);
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 *   1.1.1.   Extract   Subclass   (Interface):   DataTypeChangeable   extends
Changeable
 * and setTxMode methods are pushed down.
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface and Changeable are in-lined into
 * DataTypeChangeable. 1.6.2. Inheritance relationship is removed it is not
 * used.
 */
public interface DataTypeChangeable {

       /**
        *   1.1.2.   changeRemoteWorkingDir   method   is   pushed   down   from
Changeable.
        */
       /**
        * Sets the Tx mode either to "ascii" or to "binary"
        *
        * @param A
        *            java.lang.String containing the wished Tx-mode (Either
```

```
         *            'binary' or 'ascii')
         *@return true if succeeded; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean setTxMode(String nm);

        /**
         *  1.1.2.  changeRemoteWorkingDir  method  is  pushed  down  from
Changeable.
         */
        /**
         * Sets the Tx mode either to "ascii" or to "binary"
         *
         * @param A
         *              char containing the wished Tx-mode (Either 'i'/'b' for
binary
         *            or 'a' for ascii)
         *@return true if succeeded; otherwise false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean setTxMode(char nm);
}

/**
 * Refactoring JFTP to Framework
 * Faisal Mohammed Banaeamah - fmb@kfupm.edu.sa
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 *  1.1.1.  Extract  Subclass  (Interface):  DirectoryChangeable  extends
Changeable
 * and changeRemoteWorkingDir and changeRemoteWorkingDir methods are pushed
 * down.
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface and DirectoryChangeable are in-
lined
 * into DataTypeChangeable. 1.6.2. Inheritance relationship is removed it is
not
 * used.
 */
public interface DirectoryChangeable {

        /**
         *  1.1.2.  changeRemoteWorkingDir  method  is  pushed  down  from
Changeable.
         */
        /**
         * Changes the remote working dir.
         *
         * @param A
         *            java.lang.String containing the new path.
         *@return true if succeeded; else false.
         *@since 0.1.99t4
         *@version 1
         */
        public boolean changeRemoteWorkingDir(String newPath);
```

```
	/**
	 * 1.1.2. changeLocalWorkingDir method is pushed down from Changeable.
	 */
	/**
	 * Changes the local working dir.
	 *
	 * @param A
	 *            java.lang.String containing the new path.
	 *@return true if succeeded; else false.
	 *@since 0.1.99t4
	 *@version 1
	 */
	public boolean changeLocalWorkingDir(String newPath);
}

/*
 * Name:    Listable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 06.05.2001 Tobias Kranz Rectifing comments
 * 0.1.99t2 17.07.2001 Tobias Kranz Added 4 methods.
 * 0.1.99t4 19.07.2001 Tobias Kranz Added 2 ~'printWorkingDir' methods.
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 * Defines methods to list directories.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface is in-lined into Listable.
 * 1.6.2. Inheritance relationship is removed because it is not used.
 */
public interface Listable {
	/**
	 * Prints the current working directory on the server.
	 *
	 * @return A java.lang.String containing the directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printRemoteWorkingDir();

	/**
	 * Prints the current working directory on the local host.
	 *
	 * @return A java.lang.String containing the directory
	 *@since 0.1.99t4
	 *@version 1
	 */
	public String printLocalWorkingDir();

	/**
	 * Returns a detailed directory listing of the active directory.
	 *
```

```
         * @return Directorylist of the active directory
         *@since 0.0.2pA1
         */
        public String getLongDirList();

        /**
         * Returns a short directory listing of the active directory.
         *
         * @return Directorylist of the active directory
         *@since 0.0.2pA1
         */
        public String getShortDirList();

        /**
         * Returns a short directory listing of the active directory in a
         * jav.lang.String array.
         *
         * @return A java.lang.String array containing the Directorylist of
the
         *         active directory
         *@since 0.1.99t2
         *@version 1
         */
        public String[] getDirListArray();

        /**
         * Returns a detailed directory listing of the active working-
directory.
         *
         * @return Directorylist of the active working-directory
         *@since 0.0.2pA1
         */
        public String getLocalLongDirList();

        /**
         * Returns a short directory listing of the active working-directory.
         *
         * @return Directorylist of the active working-directory
         *@since 0.0.2pA1
         */
        public String getLocalShortDirList();

        /**
         * Returns a short directory listing of the active working-directory
in a
         * jav.lang.String array.
         *
         * @return A java.lang.String array containing the Directorylist of
the
         *         active working-directory
         *@since 0.1.99t2
         *@version 1
         */
        public String[] getLocalDirListArray();
}

/*
 * Name:    Removeable.java  [INTERFACE] (extends JFTPSuperInterface)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99t4 19.07.2001 Tobias Kranz Creation
```

```
 */
/**
 * 2.1. Extract Package: interfaces package.
 */
package jftplevelrefactoring.looplevel.interfaces;

/**
 * Defines methods to remove directories and files.
 *
 * @since 0.1.99t4
 */
/**
 * 1.6.1. Inline Class: JFTPSuperInterface is in-lined into Removeable.
 * 1.6.2. Inheritance relationship is removed because it is not used.
 */
public interface Removeable {
       /**
        * Removes a Directory on the server with the given name
        *
        * @return True if successfully removed; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean removeRemoteDir(String dirName);

       /**
        * Removes a Directory on the local fs with the given name
        *
        * @return True if successfully removed; otherwise false.
        *@since 0.1.99t4
        *@version 1
        */
       public boolean removeLocalDir(String dirName);
}

/*
 *  Name:        NetReader.java  (extends  NetIO  (extends  JFTPIO  (extends
JFTPSuper)))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.1    12.03.2001 Tobias Kranz creation.
 *          13.03.2001 Tobias Kranz changed from Thread to regular class
 *                                  (nearly total rewrite).
 * 0.0.1pA3 15.03.2001 Tobias Kranz splitted from JFTP; nothing big.
 * 0.0.1pA6 03.04.2001 Tobias Kranz added comments.
 * 0.0.1pA7 04.04.2001 Tobias Kranz splitted 'read()' to readLine() and
 *                                  readBlock().
 * 0.0.2pA2 03.05.2001 Tobias Kranz (re)added 'read()' which implements now
 *                                  'readLine()' & 'readBlock()'
 *   0.0.2pA4   15.05.2001   Tobias   Kranz   removed   everything!   created
'readStream()'
 *                                   which must be wrapped by 'receiveLine()'
&&
 *                                  'receiveBlock()' in FTPCmdServer.java .
 *                                  See README.DEVELOPMENT for details.
 * 0.1.99   10.06.2001 Tobias Kranz added 'getInputStream()'
 */
/**
 * 2.1. Extract Package: io package.
 */
package jftplevelrefactoring.looplevel.io;
```

```
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Socket;

/**
 * Class NetReader may be used to read any kind of data through an existing
 * Socket.
 *
 * @since 0.0.1
 */
public class NetReader extends NetIO {
	private InputStream inStream;

	/*
	 * private StdOut stdout = new StdOut(); private StdErr stderr = new
	 * StdErr();
	 */

	/**
	 * Sets the InputStream.
	 *
	 * @param socket
	 *               Socket to listen on (This does NOT mean a listening
socket!).
	 *@since 0.0.1
	 */
	public NetReader(Socket socket) {
		try {
			this.inStream = socket.getInputStream();
		} catch (Exception e) {
			stderr.println("Unable to get InputStream from
Socket.");
			stdout.println("Unable to get InputStream from
Socket.");
		}
	}

	/**
	 * 4.4. Pull Up Method: getStderr and setStderr methods are pulled up
to
	 * FTPConnection.
	 */
	/**
	 * 3.2. Encapsulate Fields: getStderr and setStderr.
	 */

	/**
	 * @return the stderr
	 */
	/*
	 * protected StdErr getStderr() { return stderr; }
	 */
	/**
	 * @param stderr
	 *            the stderr to set
	 */
	/*
	 * protected void setStderr(StdErr stderr) { this.stderr = stderr; }
	 */

	/*
	 * Get the stuff in.
```

```
         */
        /**
         * Autodetects the number of lines to read and give them back.
         *
         * @return String read from server
         *@since 0.0.2pA2
         */
        /*
         * public String read() { String rc = "", line; int i;
         *
         * InputStreamReader isr = new InputStreamReader(inStream);
         *
         * try { do // start reading the whole block (or just a line..) { line
= "";
         *
         * do // beginn reading the line { i = isr.read();/? should we make
         * 'isr.ready()' call(s) ? line += (char)i; } while (i != 10 ); //
until
         * '(char)10' (means CR (0x0A)) is read
         *
         * rc += line; // adding the read line to the return value
         *
         * } while ((char)line.charAt(3) != ' '); } catch (IOException ioe)
{ /
         * shall we do something here or should we throw this Exception ?? }
         *
         * return( rc.substring(0, (rc.length() - 1)) ); // trim last char
('\n') }
         */
        /**
         * Returns a InputStreamReader reading from the InputStream of the
given
         * Socket.
         *
         * @return InputStreamReader
         *@since 0.0.2pA4
         */
        public InputStreamReader getStream() {
               return (new InputStreamReader(inStream));
        }

        /**
         * Returns the InputStream of the used socket
         *
         * @return An InputStream
         *@since 0.1.99
         *@version 0
         */
        public InputStream getInputStream() {
               return (inStream);
        }

        /**
         * Reads a single line of data from the server .
         *
         * @return Whatever the ftpserver sends us.
         *@since 0.0.1
         *@deprecated
         */
        /*
         * public String readLine() { return( this.read() ); } // just to stay
         * compatible
         */
```

```
       /**
        * Reads a whole block of data from the server.
        *
        * @return Whatever the server sends.
        *@since 0.0.1
        *@deprecated
        */
       /*
        * public String readBlock() { String rc = ""; boolean done = false;
int i;
        * //char read from server
        *
        * try { InputStreamReader isr = new InputStreamReader(inStream);
        *
        * while (! done) { i = isr.read();
        *
        * if (i == -1) // -1 ~ EOF { if (jftpSuper.getDebugLevel() >= 2)
        * stdout.println("finished reading block"); done = true; } else { rc
+=
        *      (char)i;      if      (jftpSuper.getDebugLevel()      >=      2)
stdout.println("Reading " +
        * rc); } } } catch (Exception e) {
        *       stderr.println("Exception       while       reading       from
server."+e.toString()); }
        *
        * return rc; }
        */
}

/*
 *  Name:        NetWriter.java  (extends  NetIO  (extends  JFTPIO  (extends
JFTPSuper)))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.1    12.03.2001 Tobias Kranz Creation
 *          13.03.2001 Tobias Kranz Changed from Thread to regular class
 *                                  (nearly total rewrite)
 * 0.0.1pA3 15.03.2001 Tobias Kranz splitted from JFTP; nothing big
 * 0.1.99   10.06.2001 Tobias Kranz added 'getOutputStream()'
 */
/**
 * 2.1. Extract Package: io package.
 */
package jftplevelrefactoring.looplevel.io;

import java.io.OutputStream;
import java.net.Socket;

/**
 * Class NetWriter may be used to send any kind of data through an existing
 * Socket.
 *
 * @since 0.0.1
 */

/**
 * 1.6.2. Inheritance relationship is removed due to its limited usage.
1.6.2.
 * Inheritance relationship is removed and replaced by debugLevel local
 * variable.
```

```
 */
public class NetWriter extends NetIO {
	private OutputStream outStream;
	/*
	 * private StdOut stdout = new StdOut(); private StdErr stderr = new
	 * StdErr();
	 */

	/**
	 * 1.6.2. Instance variable replaces inheritance relationship.
	 */
	private short debugLevel;

	/**
	 * 1.6.3. NetWriter constructors.
	 */
	public NetWriter() {
		this.debugLevel = 0;
	}

	public NetWriter(short debugLevel) {
		this.debugLevel = debugLevel;
	}

	/**
	 * Sets the OutputStream.
	 *
	 * @param socket
	 *            Socket to write through.
	 *@since 0.0.1
	 */
	public NetWriter(Socket socket) {
		this.debugLevel = 0;
		try {
			this.outStream = socket.getOutputStream();
		} catch (Exception e) {
		}
	}

	public NetWriter(Socket socket, short debugLevel) {
		this.debugLevel = debugLevel;
		try {
			this.outStream = socket.getOutputStream();
		} catch (Exception e) {
		}
	}

	/**
	 * 4.4. Pull Up Method: getStdout and setStdout methods are pulled up
to
	 * FTPConnection.
	 */
	/**
	 * 3.2. Encapsulate Fields: getStdout and setStdout.
	 */

	/**
	 * @return the stdout
	 */
	/*
	 * protected StdOut getStdout() { return stdout; }
	 */
```

```
	/**
	 * @param stdout
	 *            the stdout to set
	 */
	/*
	 * protected void setStdout(StdOut stdout) { this.stdout = stdout; }
	 */

	/**
	 * 4.4. Pull Up Method: getStderr and setStderr methods are pulled up
to
	 * FTPConnection.
	 */
	/**
	 * 3.2. Encapsulate Fields: getStderr and setStderr.
	 */

	/**
	 * @return the stderr
	 */
	/*
	 * protected StdErr getStderr() { return stderr; }
	 */

	/**
	 * @param stderr
	 *            the stderr to set
	 */
	/*
	 * protected void setStderr(StdErr stderr) { this.stderr = stderr; }
	 */

	/**
	 * Returns the OutputStream of the used socket
	 *
	 * @return The OutputStream used for the current socket.
	 *@since 0.1.99
	 *@version 0
	 */
	public OutputStream getOutputStream() {
		return (outStream);
	}

	/*
	 * Just send everything out to the server
	 */
	/**
	 * Writes data to the ftpserver
	 *
	 * @param string
	 *            The data to send
	 *@since 0.0.1
	 */
	public void write(String string) {
		int i;

		try {
			if (debugLevel >= 1)
				getStdout().println("starting new write-loop");

			for (i = 0; i < string.length(); i++) {
				outStream.write(string.charAt(i));
				if (debugLevel >= 2)
```

```
                                    getStdout().println("writing   char:   "   +
string.charAt(i));
                    }
                    if (string.charAt(string.length() - 1) != '\n')
                          outStream.write('\n'); // Finish the line :-)
                    else if (debugLevel >= 2)
                          getStderr().println("DON'T SEND NEWLINES!");

                    outStream.flush(); // make sure the chars are written

                    if (debugLevel >= 1)
                          getStdout().println("ending write-loop");
             } catch (Exception e) {
                    getStderr().println("Exception  in  Outp("  +  e.toString()
+ ")");
             }
       }
}

/*
 * Name:    StdErr.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * 2.1. Extract Package: io package.
 */
package jftplevelrefactoring.looplevel.io;

/**
 * Represents the standard error channel (StdErr).<BR>
 * The trailing '*' will be removed in stable versions.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: JFTPIO and StdIO are in-lined into StdErr and
 * inheritance relationship is directly moved to StdErr. 1.6.2. Inheritance
 * relationship is removed because it is not used.
 */
public class StdErr {
       /**
        * Prints an object to StdErr.
        *
        * @param o
        *            Object to print
        *@since 0.0.2pA1
        */
       public void print(Object o) {
             System.err.print("*" + o);
       }

       /**
        * Prints a double to StdErr.
        *
        * @param d
        *            double to print
        *@since 0.0.2pA1
        */
       public void print(double d) {
```

```
		System.err.print("*" + d);
	}

	/**
	 * Prints a float to StdErr.
	 *
	 * @param f
	 *            float to print
	 *@since 0.0.2pA1
	 */
	public void print(float f) {
		System.err.print("*" + f);
	}

	/**
	 * Prints a long to StdErr.
	 *
	 * @param l
	 *            long to print
	 *@since 0.0.2pA1
	 */
	public void print(long l) {
		System.err.print("*" + l);
	}

	/**
	 * Prints an int to StdErr.
	 *
	 * @param i
	 *            int to print
	 *@since 0.0.2pA1
	 */
	public void print(int i) {
		System.err.print("*" + i);
	}

	/**
	 * Prints a short to StdErr.
	 *
	 * @param s
	 *            short to print
	 *@since 0.0.2pA1
	 */
	public void print(short s) {
		System.err.print("*" + s);
	}

	/**
	 * Prints a char to StdErr.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void print(char c) {
		System.err.print("*" + c);
	}

	/**
	 * Prints a byte to StdErr.
	 *
	 * @param b
	 *            byte to print
```

```
         *@since 0.0.2pA1
         */
        public void print(byte b) {
               System.err.print("*" + b);
        }

        /**
         * Prints a boolean to StdErr.
         *
         * @param b
         *            boolean to print
         *@since 0.0.2pA1
         */
        public void print(boolean b) {
               System.err.print("*" + b);
        }

        /**
         * Prints an object and "\n\r" to StdErr.
         *
         * @param o
         *            Object to print
         *@since 0.0.2pA1
         */
        public void println(Object o) {
               System.err.println("*" + o);
        }

        /**
         * Prints a double and "\n\r" to StdErr.
         *
         * @param d
         *            double to print
         *@since 0.0.2pA1
         */
        public void println(double d) {
               System.err.println("*" + d);
        }

        /**
         * Prints a float and "\n\r" to StdErr.
         *
         * @param f
         *            float to print
         *@since 0.0.2pA1
         */
        public void println(float f) {
               System.err.println("*" + f);
        }

        /**
         * Prints a long and "\n\r" to StdErr.
         *
         * @param l
         *            long to print
         *@since 0.0.2pA1
         */
        public void println(long l) {
               System.err.println("*" + l);
        }

        /**
         * Prints a int and "\n\r" to StdErr.
```

```
         *
         * @param i
         *            int to print
         *@since 0.0.2pA1
         */
        public void println(int i) {
               System.err.println("*" + i);
        }

        /**
         * Prints a short and "\n\r" to StdErr.
         *
         * @param s
         *            short to print
         *@since 0.0.2pA1
         */
        public void println(short s) {
               System.err.println("*" + s);
        }

        /**
         * Prints a char and "\n\r" to StdErr.
         *
         * @param c
         *            char to print
         *@since 0.0.2pA1
         */
        public void println(char c) {
               System.err.println("*" + c);
        }

        /**
         * Prints a byte and "\n\r" to StdErr.
         *
         * @param b
         *            byte to print
         *@since 0.0.2pA1
         */
        public void println(byte b) {
               System.err.println("*" + b);
        }

        /**
         * Prints a boolean and "\n\r" to StdErr.
         *
         * @param b
         *            boolean to print
         *@since 0.0.2pA1
         */
        public void println(boolean b) {
               System.err.println("*" + b);
        }
}

/*
 * Name:    StdIn.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.1.99t5 01.08.2001 Tobias Kranz added 'readPassword()'
 */
```

```
/**
 * 2.1. Extract Package: io package.
 */
package jftplevelrefactoring.looplevel.io;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * Represents the standard input channel (StdIn).
 *
 * @since 0.0.2pA1
 */
/**
 *  1.6.1.  Inline  Class:  JFTPIO  and  StdIO  are  in-lined  into  StdIn  and
inheritance
 * relationship is directly moved to StdIn. 1.6.2. Inheritance relationship
is
 * removed because it is not used.
 */
public class StdIn {
	private BufferedReader br;

	public StdIn() {
		this.br = new BufferedReader(new InputStreamReader(System.in));
	}

	/**
	 * Reads a whole line of data from the standard input channel.
	 *
	 * @return The String beeing read.
	 *@throws IOException
	 *             if reading fails.
	 *@since 0.0.2pA1
	 */
	public String readLine() throws IOException {
		return ((String) br.readLine());
	}

	/**
	 * Reads a single char of data from the standard input channel.
	 *
	 * @return The char beeing read.
	 *@throws IOException
	 *             if reading fails.
	 *@since 0.0.2pA1
	 */
	protected char readChar() throws IOException {
		return ((char) br.read());
	}

	/**
	 * Reads an int from the standard input channel.
	 *
	 * @return The int being read
	 *@throws IOException
	 *             if reading fails
	 *@since 0.0.2pA1
	 */
	protected int readInt() throws IOException {
```

```
                return (Integer.valueOf(br.readLine()).intValue());
        }

        /**
         * Reads an InetAddress from the standard input channel.
         *
         * @return The InetAddress read
         *@throws IOException
         *             if reading fails
         *@throws UnknownHostException
         *             if host is unknown
         *@since 0.0.2pA1
         */
        protected InetAddress readInetAddress() throws IOException,
                    UnknownHostException {
                return (InetAddress.getByName(br.readLine()));
        }

        /**
         * Reads a password and dissables the terminal-echo.
         *
         * @return The Password read from StdIn
         *@throws IOException
         *             if reading fails
         *@since 0.1.99t5
         *@version 1
         */
        public String readPasswd() throws IOException {
                String rc = "";
                StringBuffer sb = new StringBuffer();
                // char c;//////////////////////////////////////

                /*
                 * Font-Colors:
                 *
                 *  Font-Colors are from 29 - 39. Background-Colors beginning
with 40.
                 *
                 *  #  color  29  Gray  30  Black  31  DarkRed  32  DarkGreen  33
Brown/DarkOrange
                 * 34 DarkBlue 35 Purple 36 Marine 37 Gray 38 White 39 Gray
                 */

                // Set Font-Color to Black
                sb.append('\u001b');
                sb.append('[');
                sb.append(30); // Color
                sb.append('m');
                System.out.print(sb.toString()); // Set it.

                // read
                rc = br.readLine();

                // (re)Set Font-Color to Gray
                sb = new StringBuffer(); // Reset it
                sb.append('\u001b');
                sb.append('[');
                sb.append(39); // Color
                sb.append('m');
                System.out.print(sb.toString()); // Set it.

                return (rc);
        }
```

```
}

/*
 * Name:    StdOut.java (extends StdIO (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 * 0.0.2pA4 15.05.2001 Tobias Kranz fixed "double newline"-bug
 */
/**
 * 2.1. Extract Package: io package.
 */
package jftplevelrefactoring.looplevel.io;

/**
 * Represents the standard output channel (StdOut)
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: JFTPIO and StdIO are in-lined into StdOut and
 * inheritance relationship is directly moved to StdOut. 1.6.2. Inheritance
 * relationship is removed and replaced by isCli local variable.
 */
public class StdOut {

	/**
	 * 1.6.2. Instance variable replaces inheritance relationship.
	 */
	boolean isCli;

	/**
	 * 1.6.3. StdOut constructors.
	 */
	public StdOut() {
		this.isCli = false;
	}

	public StdOut(boolean isCli) {
		this.isCli = isCli;
	}

	/**
	 * Prints an object to StdOut.
	 *
	 * @param o
	 *             Object to print
	 *@since 0.0.2pA1
	 */
	public void print(Object o) {
		if (isCli) {
			System.out.print(o);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a double to StdOut.
	 *
	 * @param d
```

```
         *            double to print
         *@since 0.0.2pA1
         */
        public void print(double d) {
               if (isCli) {
                      System.out.print(d);
               } else {
                      // Insert GUI-Code here
               }
        }

        /**
         * Prints a float to StdOut.
         *
         * @param f
         *            float to print
         *@since 0.0.2pA1
         */
        public void print(float f) {
               if (isCli) {
                      System.out.print(f);
               } else {
                      // Insert GUI-Code here
               }
        }

        /**
         * Prints a long to StdOut.
         *
         * @param l
         *            long to print
         *@since 0.0.2pA1
         */
        public void print(long l) {
               if (isCli) {
                      System.out.print(l);
               } else {
                      // Insert GUI-Code here
               }
        }

        /**
         * Prints an int to StdOut.
         *
         * @param i
         *            int to print
         *@since 0.0.2pA1
         */
        public void print(int i) {
               if (isCli) {
                      System.out.print(i);
               } else {
                      // Insert GUI-Code here
               }
        }

        /**
         * Prints a short to StdOut.
         *
         * @param s
         *            short to print
         *@since 0.0.2pA1
         */
```

```
	public void print(short s) {
		if (isCli) {
			System.out.print(s);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a byte to StdOut.
	 *
	 * @param b
	 *            byte to print
	 *@since 0.0.2pA1
	 */
	public void print(byte b) {
		if (isCli) {
			System.out.print(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a char to StdOut.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void print(char c) {
		if (isCli) {
			System.out.print(c);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a boolean to StdOut.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void print(boolean b) {
		if (isCli) {
			System.out.print(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints an object and "\n\r" to StdOut.
	 *
	 * @param o
	 *            Object to print
	 *@since 0.0.2pA1
	 *@version 0
	 */
	public void println(Object o) {
		if (isCli) {
```

```
                        if  ((char)  o.toString().charAt(o.toString().length()  -
1) == '\n')
                                System.out.print(o);
                        else
                                System.out.println(o);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints a double and "\n\r" to StdOut.
         *
         * @param d
         *             double to print
         *@since 0.0.2pA1
         */
        public void println(double d) {
                if (isCli) {
                        System.out.println(d);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints a float and "\n\r" to StdOut.
         *
         * @param f
         *             float to print
         *@since 0.0.2pA1
         */
        public void println(float f) {
                if (isCli) {
                        System.out.println(f);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints a long and "\n\r" to StdOut.
         *
         * @param l
         *             long to print
         *@since 0.0.2pA1
         */
        public void println(long l) {
                if (isCli) {
                        System.out.println(l);
                } else {
                        // Insert GUI-Code here
                }
        }

        /**
         * Prints an int and "\n\r" to StdOut.
         *
         * @param i
         *             int to print
         *@since 0.0.2pA1
         */
        public void println(int i) {
```

```
		if (isCli) {
			System.out.println(i);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a short and "\n\r" to StdOut.
	 *
	 * @param s
	 *            short to print
	 *@since 0.0.2pA1
	 */
	public void println(short s) {
		if (isCli) {
			System.out.println(s);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a byte and "\n\r" to StdOut.
	 *
	 * @param b
	 *            byte to print
	 *@since 0.0.2pA1
	 */
	public void println(byte b) {
		if (isCli) {
			System.out.println(b);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a char and "\n\r" to StdOut.
	 *
	 * @param c
	 *            char to print
	 *@since 0.0.2pA1
	 */
	public void println(char c) {
		if (isCli) {
			System.out.println(c);
		} else {
			// Insert GUI-Code here
		}
	}

	/**
	 * Prints a boolean and "\n\r" to StdOut.
	 *
	 * @param b
	 *            boolean to print
	 *@since 0.0.2pA1
	 */
	public void println(boolean b) {
		if (isCli) {
			System.out.println(b);
		} else {
```

```
                    // Insert GUI-Code here
                }
        }
}

/*
 * Name:    NetIO.java (extends JFTPIO)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * 2.1. Extract Package: io package.
 */
package jftplevelrefactoring.looplevel.io;

/**
 * Class for network-io.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: JFTPIO is in-lined into NetIO and inheritance
 * relationship is directly moved to NetIO. 1.6.2. Inheritance relationship
is
 * removed due to its limited usage.
 */
public class NetIO {
        /**
         * 3.5. Pull Up Field: stdout and stderr are pulled up to NetIO.
         */
        protected StdOut stdout = new StdOut();
        protected StdErr stderr = new StdErr();

        /**
         * 4.4. Pull Up Method: getStdout and setStdout methods are pulled up
to
         * NetIO.
         */
        /**
         * 3.2. Encapsulate Fields: getStdout and setStdout.
         */

        /**
         * @return the stdout
         */
        protected StdOut getStdout() {
                return stdout;
        }

        /**
         * @param stdout
         *            the stdout to set
         */
        protected void setStdout(StdOut stdout) {
                this.stdout = stdout;
        }

        /**
         * 4.4. Pull Up Method: getStderr and setStderr methods are pulled up
to
```

```
         * NetIO.
         */
        /**
         * 3.2. Encapsulate Fields: getStderr and setStderr.
         */
        /**
         * @return the stderr
         */
        protected StdErr getStderr() {
              return stderr;
        }

        /**
         * @param stderr
         *            the stderr to set
         */
        protected void setStderr(StdErr stderr) {
              this.stderr = stderr;
        }
}

/*
 * Name:    CommandLineParser.java (extends Parser (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation.
 * 0.0.2pA2 03.05.2001 Martin Loh   GET implemented
 * 0.0.2pA4 15.05.2001 Tobias Kranz fixed 'containsWildcards(String, int)'
&&
 *                                  'parse(String)'
 * 0.0.2pA4 05.06.2001 Tobias Kranz added 'license' command
 * 0.1.99   10.06.2001 Tobias Kranz added possibility to set Tx modes.
 * 0.1.99t2 17.07.2001 Tobias Kranz 2nd fix of 'containsWildcards(String,
int)'
 *                                  added 'createPattern(String)' &&
 *                                            'comparePatternWithList(String,
String[])',
 *                                  implemented MGET and MPUT and the
 *                                  "local dir list" command-group.
 * 0.1.99t4 19.07.2001 Tobias Kranz added 'void pager(String)' && fixed bug
with
 *                                                    MGET && added 'String
getKnownCommands()' &&
 *                                       implemented MKDIR && LMKDIR && PWD &&
LPWD
 * 0.1.99t5 23.07.2001 Tobias Kranz improved lcd
 */
/**
 * 2.1. Extract Package: parsers package.
 */
package jftplevelrefactoring.looplevel.parsers;

import java.io.IOException;
import java.util.NoSuchElementException;
import java.util.StringTokenizer;
import java.util.Vector;

import jftplevelrefactoring.looplevel.core.FTPASCIIData;
import jftplevelrefactoring.looplevel.core.FTPAuthentication;
import jftplevelrefactoring.looplevel.core.FTPBinaryData;
import jftplevelrefactoring.looplevel.core.FTPConnection;
```

```
import jftplevelrefactoring.looplevel.core.FTPDataType;
import jftplevelrefactoring.looplevel.core.FTPDirectory;
import jftplevelrefactoring.looplevel.core.FTPTransfer;
import jftplevelrefactoring.looplevel.io.StdErr;
import jftplevelrefactoring.looplevel.io.StdIn;
import jftplevelrefactoring.looplevel.io.StdOut;

/**
 * Parses the CommandLine for known FTP-Commands.<BR>
 * This class is used by JFTP.<BR>
 *
 * @since 0.0.2pA1
 */
/**
 * 1.1.3.2. A set of local variables of JFTPSuper to cover jftpSuper.
 */
public class CommandLineParser extends Parser {
	private final char[] listLong1 = { 'l' };
	private final char[] listLong2 = { 'l', 's', ' ', '-', 'l' };
	private final char[] listShort1 = { 'l', 's' };
	private final char[] listShort2 = { 'd', 'i', 'r' };
	// the same for local
	private final char[] llistLong1 = { 'l', 'l' };
	private final char[] llistLong2 = { 'l', 'l', 's', ' ', '-', 'l' };
	private final char[] llistShort1 = { 'l', 'l', 's' };
	private final char[] llistShort2 = { 'l', 'd', 'i', 'r' };

	/**
	 *   1.1.3.2.   A   set   of   local   variables   of   FTPConnection,
FTPAuthentication,
	 * FTPTransfer, FTPDirectory and FTPDataType.
	 */
	private FTPConnection ftpConnection;
	private FTPAuthentication ftpAuthentication;
	private FTPTransfer ftpTransfer;
	private FTPDirectory ftpDirectory;
	private FTPDataType ftpASCIIData;
	private FTPDataType ftpBinaryData;

	// private FTPCmdServer fcs;

	/**
	 * Constructor
	 *
	 * @since The beginning of Time.
	 */
	public CommandLineParser() {
		// this.fcs = new FTPCmdServer();
		this.ftpConnection = new FTPConnection();
		this.ftpAuthentication = new FTPAuthentication();
		this.ftpTransfer = new FTPTransfer();
		this.ftpDirectory = new FTPDirectory();
		this.ftpASCIIData = new FTPASCIIData();
		this.ftpBinaryData = new FTPBinaryData();
	}

	public CommandLineParser(FTPConnection ftpConnection) {
		this.ftpConnection = ftpConnection;
		this.ftpAuthentication = new FTPAuthentication(ftpConnection
				.getServer(), ftpConnection.getServerPort());
		this.ftpTransfer = new FTPTransfer(ftpConnection.getServer(),
				ftpConnection.getServerPort());
		this.ftpDirectory = new FTPDirectory(ftpConnection.getServer(),
```

```
                        ftpConnection.getServerPort());
            this.ftpASCIIData = new FTPASCIIData(ftpConnection.getServer(),
                        ftpConnection.getServerPort());
            this.ftpBinaryData                    =                    new
FTPBinaryData(ftpConnection.getServer(),
                        ftpConnection.getServerPort());
      }

      /**
       * 4.8. Remove Assignments to Parameters: assignments of parameter of
parse
       * method are removed by using temporary variable.
       */
      /**
       * Parses the CommandLine.
       *
       * @param s
       *            String to parse.
       *@version 6
       *@since 0.0.2pA1
       */
      public void parse(String s) {
            try {

                  /*
                   * Have you ever wondered how the parsing works? There
are 3 main
                   * parts: At the first point we are checking for
commands that can
                   * be entered in any situation. At the second point we
are looking
                   * for commands that only makes sense if we're
connected. And at
                   * least there's a section at the end where all command
are handled
                   * which may entered in an unconnected status.
                   */

                  /**
                   * 4.8.1. Temporary variable is set value of parameter.
4.8.2. All
                   * assignments to parameter are set to temporary
variable.
                   */
                  /**
                   * 7.2. Rename Variable: temp variable is renamed to
command.
                   */
                  String command = s;
                  command = command.trim();

                  if (ftpConnection.getDebugLevel() >= 2)
                        printErrorLine("parsing \"" + command + "\"");

                  // OPENs a connection
                  if (isOpenCommand(command)) {
                        this.logon(command);
                  } else if (isInfoCommand(command)) {
                        ftpConnection.printLongJFTPInfo();
                  } else if (isLicenseCommand(command)) {
                        pager(ftpConnection.getLicense());
                  } else if (isQuitExitCommand(command)) {
                        printOutputLine("obsolete. Use 'bye' instead.");
```

```
                    }
                    // LocalLiSt short
                    else if (isShortLocalListCommand(command)) {
                          printOutput(ftpDirectory.getLocalShortDirList());
                    }
                    // LocalLiSt Long
                    else if (isLongLocalListCommand(command)) {
                          printOutput(ftpDirectory.getLocalLongDirList());
                    }
                    // HELP
                    else if (isHelpCommand(command)) {
                          String tmp;
                          tmp = getKnownCommands();

                          if (ftpConnection.isConnected()) {
                                tmp += "Server Commands:\n";
                                ftpTransfer.send(command); // Sends 'help'
to the server
                                tmp += ftpTransfer.readCtrl(); // Reads the
servers
                                // commands.
                          }
                          pager(tmp);
                    }
                    // LMKDIR
                    else if (isMakeLocalDirectoryCommand(command)) {
                          if
(ftpDirectory.createLocalDir(command.substring(7)))
                                printOutputLine("Okay, directory \"" +
command.substring(7)
                                            + "\" was created.");
                          else
                                printOutputLine("Unable to create a
directory.");
                    }
                    // LRMDIR
                    else if (isRemoveLocalDirectoryCommand(command)) {
                          if
(ftpDirectory.removeLocalDir(command.substring(7)))
                                printOutputLine("Okay, directory \"" +
command.substring(7)
                                            + "\" was removed.");
                          else
                                printOutputLine("Unable to remove that
directory.");
                    }
                    // LPWD
                    else if (isPrintLocalDirectoryCommand(command)) {

      printOutputLine(ftpDirectory.printLocalWorkingDir());
                    }
                    // LCD
                    else if (isChangeLocalDirectoryCommand(command)) {
                          if
(!ftpDirectory.changeLocalWorkingDir(command.substring(4)))
                                printOutputLine("Can't change working dir
to: "
                                            + command.substring(4));
                          else
                                printOutputLine("Okay, working dir
changed.");
                    }
```

```
                    //  The  following  if's  are  only  interresting  if  we're
connected
                    else if (ftpConnection.isConnected()) {
                           // CLOSEs a connection
                           if (isCloseCommand(command)) {
                                  ftpConnection.disconnect();
                           }
                           // Change working Directory
                           else if (isChangeDirectoryCommand(command)) {

      ftpDirectory.changeRemoteWorkingDir(command.substring(3)
                                                .trim());
                           }
                           // LiSt short
                           else if (isShortListCommand(command)) {

      printOutput(ftpDirectory.getShortDirList());
                           }
                           // LiSt Long
                           else if (isLongListCommand(command)) {
                                  printOutput(ftpDirectory.getLongDirList());
                           }
                           // GET
                           else if (isGetCommand(command)) {
                                  if       (containsWildcards(command,       4,
command.length())) {
                                        printOutputLine("Wildcards   are   NOT
allowed. Use 'mget' instead.");
                                  } else {
                                        if
(!ftpTransfer.getFile(command.substring(4).trim(),

      command.substring(4).trim()))
                                                printErrorLine("Unable  to  get
the file.");
                                  }
                           }
                           // MGET
                           else if (isMGetCommand(command)) {
                                  String[]                tmp                =
comparePatternWithList(createPattern(command
                                                .substring(4).trim()),
ftpDirectory
                                                .getDirListArray());
                                  ftpTransfer.getFiles(tmp, tmp);
                           }
                           // PUT
                           else if (isPutCommand(command)) {
                                  if       (containsWildcards(command,       4,
command.length())) {
                                        printOutputLine("Wildcards   are   NOT
allowed. Use 'mput' instead.");
                                  } else {
                                        if
(!ftpTransfer.putFile(command.substring(4)))
                                                printOutputLine("Unable to put
the file.");
                                  }
                           }
                           // MPUT
                           else if (isMPutCommand(command)) {
                                  String[]                tmp                =
comparePatternWithList(createPattern(command
```

```
                                                .substring(4).trim()),
ftpDirectory
                                                .getLocalDirListArray());
                                    ftpTransfer.putFiles(tmp);
                              }
                              // setting the TX mode
                              else if (isTypeCommand(command)) {
                                    if (command.equals("bin"))
                                          ftpBinaryData.setTxMode(command);
                                    if (command.equals("asc"))
                                          ftpASCIIData.setTxMode(command);
                              }
                              // MKDIR
                              else if (isMakeDirectoryCommand(command)) {

      ftpDirectory.createRemoteDir(command.substring(6));
                              }
                              // RMDIR
                              else if (isRemoveDirectoryCommand(command)) {

      ftpDirectory.removeRemoteDir(command.substring(6));
                              }
                              // PWD
                              /*
                               * Yes, this costs performance, but its easier to
change the
                               *  progx  behavior,  for  example  when  adapting
another protocol.
                               */
                              else if (isPrintDirectoryCommand(command)) {

      printOutputLine(ftpDirectory.printRemoteWorkingDir());
                              } else if (isByeCommand(command)) {
                                    ftpConnection.disconnect();
                                    System.exit(0); // EXIT(0)
                              } else {
                                    if  (command.length()  >  0)  //  prevents
errors if you just
                                    // type
                                    // '\n'
                                    {
                                          if  (ftpConnection.getDebugLevel()  >=
2)
                                                printOutputLine("sending:  "  +
command);
                                          ftpTransfer.send(command);

      printOutputLine(ftpTransfer.readCtrl());
                                    }
                              }
                        } else // At this point we are not connected:
                        {
                              if (isByeCommand(command)) {
                                    System.exit(0); // EXIT(0)
                              } else {
                                    if  (command.length()  >  0)  //  prevents
errors if you just
                                    // type
                                    // '\n'
                                    {
                                          printErrorLine("Unable  to  execute  "
+ command + " .");
                                    }
```

```
                        }
                }
            } catch (StringIndexOutOfBoundsException siobe) {
                if (ftpConnection.getDebugLevel() >= 2)
                    printErrorLine("Error in main-loop of Clp: " +
siobe.toString());
            } catch (NullPointerException npe) {
                if (ftpConnection.getDebugLevel() >= 2)
                    printOutputLine("NullPointerException caugth (in
clp)");
            } catch (NoSuchElementException nsee) {
                if (ftpConnection.getDebugLevel() >= 2)
                    printOutputLine("NoSuchElementException caugth (in
clp)");
            }
        }

        /**
         * Will be implemented when j2sdk1.4 is released!
         */
        /*
         * IS implemented! But in FTPCmdServer.\ private void getFiles(String
         * pattern) { }
         */

        /**
         * Manages the whole authentication/logon procedure
         *
         * @param s
         *            String containing all params entered
         *@since 0.0.2pA4
         *@version 2
         */
        private void logon(String s) {
            boolean serverIsValid = false;
            String server;

            /*
             * s.substring(5) will 'return false' if s.length < 5 !!!
             */
            if (s.length() > 5)
                server = s.substring(5).trim();
            else
                server = "";

            if (ftpConnection.getDebugLevel() >= 2)
                printErrorLine("server=\"" + server + "\"");

            if (ftpConnection.isConnected()) {
                printOutputLine("Unable to re-connect; You must
disconnect first...");
            } else {
                while (!serverIsValid) {
                    if (server.length() > 4)// What's the min length
of a valid inet
                    // Address?
                    {
                        serverIsValid = true;
                    } else {
                        printOutput("(host): ");
                        try {
                            server = readInputLine();
                        } catch (IOException ioe) {
```

```
                                }
                        }
                }

                if (ftpConnection.getDebugLevel() >= 2)
                        printOutputLine("contacting server " + server);

                if (ftpConnection.setServerTo(server)) // setting (new)
Server
                // (Server
                // is _really_ there...:~) )
                {
                        ftpConnection.connect(); // CONNECT !
                        if (ftpConnection.isConnected()) // connected ?
                        {
                                // if (jftpSuper.getDebugLevel() >= 2)
                                printOutputLine(ftpTransfer.readCtrl());

                                if (!ftpAuthentication.authenticate()) {
                                        /*
                                         *     Should     be     handled     by
authenticate.
                                         */
                                        //      printErrorLine("Unable     to
authenticate.");
                                        parse("close"); // close connection
from Server
                                }
                        }
                } else {
                        printErrorLine("Cannot connect to server.");
                }
        }
    }

    /**
     *   4.6.   Add   Parameter:   integer   end   parameter   is   added   to
containsWildcards
     * method.
     */
    /**
     * Checks if the given String contains any wildcards ('*'/'?').
     *
     * @param s
     *             String to check
     *@param start
     *            int position to start at.
     *@return true if the given String contains any wildcards; otherwise
not
     *@since 0.0.2pA3
     */
    private boolean containsWildcards(String s, int start, int end) {
            boolean rc = false;
            int i;

            // Check for a valid range
            if (start <= s.length()) {
                    for (i = start; i < end; i++) {
                            if   (((char)   s.charAt(i)   ==   '*')   ||   ((char)
s.charAt(i) == '?')
                                            || ((char) s.charAt(i) == '[')) {
                                    rc = true;
                                    break;
```

```
                        }
                }
        }

        return (rc);
    }

    /**
     * Creates a regular-expression-pattern usable by java.lang.util.regex
from
     * any posix-regex. Example: Posix-regex: java.lang.util.regex: (see
     * j2sdk1.4-doc) *[0-9]t.ab? {graph}*[0-9]t.ab{graph}? ||
     * {graph}*{num}?t.ab{graph}?
     *
     * @param pattern
     *            The posix-regex to compute
     *@return The matching java.lang.regex-regex
     *@since 0.1.99t2
     *@version 1
     */
    private String createPattern(String pattern) {
        // Initial things
        StringTokenizer st;
        String pat = pattern, // local-temporary pattern
        rc = "";

        if (ftpConnection.getDebugLevel() >= 2)
            printErrorLine("Pattern=" + pattern);
        /* <Parsing for '''s> */
        st = new StringTokenizer(pat, "*");

        if (st.hasMoreTokens())
            rc += st.nextToken();

        while (st.hasMoreTokens())
            rc += "{graph}*" + st.nextToken();

        // Check for start and end
        if (pat.charAt(0) == '*')
            rc = "{graph}*" + rc;

        if ((pattern.charAt(pattern.length() - 1) == '*'))
            rc += "{graph}*";
        /* </Parsing for '''s> */

        // Resetting
        st = null;
        pat = rc;
        rc = "";

        /* <Parsing for '?''s> */
        st = new StringTokenizer(pat, "?");

        if (st.hasMoreTokens())
            rc += st.nextToken();

        while (st.hasMoreTokens())
            rc += "{graph}?" + st.nextToken();

        // Check for start and end
        if (pat.charAt(0) == '?')
            rc = "{graph}?" + rc;
```

```
              if ((pattern.charAt(pattern.length() - 1) == '?'))
                     rc += "{graph}?";
              /* </Parsing for '?''s> */

              /*
               * Should be used before running another loop on the pattern.
//
               * Resetting st = null; pat = rc; rc = "";
               */
              return (rc);
       }

       /**
        * Compares a java.lang.String array with a given java.util.regex
compatible
        * regular expression and returns a list of all matching entries.
        *
        * @param pattern
        *            The java.util.regex pattern to use.
        *@param list
        *            A java.lang.String array which should be compared to the
        *            pattern
        *@return A java.lang.String array which all entries from list that
are
        *         matching the pattern.
        *@since 0.1.99t2
        *@version 1
        */
       private String[] comparePatternWithList(String pattern, String[] list)
{
              int i;
              java.util.regex.Pattern                    p                    =
java.util.regex.Pattern.compile(pattern);
              java.util.regex.Matcher m;
              Vector<String> tmp = new Vector<String>(1);

              for (i = 0; i < list.length; i++) {
                     m = p.matcher(list[i]);

                     if (m.matches())
                            tmp.add((String) list[i]);
              }

              // Vector tmp -> String rc[n]
              String[] rc = new String[tmp.size()];

              for (i = 0; i < tmp.size(); i++)
                     rc[i] = tmp.elementAt(i);

              return (rc);
       }

       /**
        * 4.8. Remove Assignments to Parameters: assignments of parameter of
pager
        * method are removed by using temporary variable.
        */
       /**
        * Show any txpe of text side by side. A pager. This method looks for
'\n''s
        * (24 times), then shows a "Press return.."-msg and waits for any
        * return-terminated input and restarts that loop until EOF.
        *
```

```
        * @param text
        *            The text to display
        *@since 0.1.99.test4
        *@version 3
        */
       private void pager(String text) {
              int i, j = 0;
              char c;

              /**
               * 4.8.1. Temporary variable is set value of parameter. 4.8.2.
All
               * assignments to parameter are set to temporary variable.
               */
              String temp = text;
              temp += "\n"; // Ugly way to make sure the prompt comes up in a
new
              // line.

              while (temp.length() > j) {
                     for (i = 0; i < 23; i++) {
                            do {
                                   c = temp.charAt(j++);
                                   printOutput(c + "");
                            } while (c != '\n');
                     }

                     printOutput("\nPress [Return] to continue...");
                     try {
                            readInputLine();
                     } catch (IOException whoCares) {
                     }
              }
       }

       /**
        * Returns a java.lang.String containing all commands known by 'this'.
        *
        * @return A java.lang.String containing all commands.
        *@since 0.1.99test4
        *@version 1
        */
       private String getKnownCommands() {
              return    ("Commands    recognized    by    JavaFTP:    (*    =>
unimplemented)\n"
                           + "\n"
                           + "Conntection-commands:\n"
                           + "\n"
                           + "open           Asks you the server's name or ip
to connect to.\n"
                           + "open SERVER    Opens a connection to the server
SERVER.\n"
                           + "close          Closes the active connection.\n"
                           + "\n"
                           + "List-commands:\n"
                           + "l\n"
                           + "ls -la            Shows a detailed list of the
active directory "
                           + "on the server.\n"
                           + "\n"
                           + "ls\n"
                           + "dir            Shows a short list of the active
directory on the server.\n"
```

```
                            + "\n"
                            + "ll\n"
                            + "lls -la          Shows a detailed list of the
active directory on "
                            + "the localhost.\n"
                            + "\n"
                            + "lls\n"
                            + "ldir           Shows a short list of the active
directory on "
                            + "the localhost.\n"
                            + "\n"
                            + "Options and switches:\n"
                            + "type X              Sets the Tx-mode to X.
(X=i(binary)||X=a(ascii))\n"
                            + "asc            Sets the Tx-mode to ascii.\n"
                            + "bin            Sets the Tx-mode to binary.\n"
                            + "\n"
                            + "Directory relative commands:\n"
                            + "cd PATH         Changes the active directory on
the server to PATH.\n"
                            + "lcd PATH        Changes the active directory on
the localhost to PATH.\n"
                            + "pwd                 Shows the current working
directory on the server.\n"
                            + "lpwd                Shows the current working
directory on the localhost.\n"
                            + "mkdir DIR      Creates a directory named DIR on
the server.\n"
                            + "rmdir DIR      Removes a directory named DIR on
the server.\n"
                            + "lmkdir DIR     Creates a directory named DIR on
the localhost.\n"
                            + "lrmdir DIR     Removes a directory named DIR on
the localhost.\n"
                            + "\n"
                            + "Transfer-commands:\n"
                            + "get FILE       Gets a file named FILE\n"
                            + "put FILE       Puts a file named FILE\n"
                            + "mget PATTERN   Gets all files that matches the
pattern PATTERN.\n"
                            + "mput PATTERN   Puts all files that matches the
pattern PATTERN.\n"
                            + "\n"
                            + "Misc stuff:\n"
                            + "info                Shows a short info about
JavaFTP.\n"
                            + "license        Shows the license under "
                            + "wich you are allowed to use JavaFTP.\n"
                            + "help           Shows this helpscreen.\n"
                            + "bye              Quits the program and closses
all open connections.\n"
                            + "\n");
      }

      /**
       *
       *  1.4.  Hide  Delegate:  StdIn,  StdOut  and  StdErr  are  hidden  by
readInputLine,
       * printOutput, printOutputLine and printErrorLine methods.
       */
      private String readInputLine() throws IOException {
            StdIn stdin = new StdIn();
            return stdin.readLine();
```

```
	}

	private void printOutput(String message) {
		StdOut stdout = new StdOut();
		stdout.print(message);
	}

	private void printOutputLine(String message) {
		StdOut stdout = new StdOut();
		stdout.println(message);
	}

	private void printErrorLine(String message) {
		StdErr stderr = new StdErr();
		stderr.println(message);
	}

	/**
	 * 4.2. Decompose Conditional: isOpenCommand method.
	 */
	private boolean isOpenCommand(String command) {
		return (command.startsWith("open"));
	}

	/**
	 * 4.2. Decompose Conditional: isInfoCommand method.
	 */
	private boolean isInfoCommand(String command) {
		return (command.equals("info"));
	}

	/**
	 * 4.2. Decompose Conditional: isLicenseCommand method.
	 */
	private boolean isLicenseCommand(String command) {
		return (command.equals("license"));
	}

	/**
	 * 4.2. Decompose Conditional: isQuitCommand method.
	 */
	/**
	 * 7.1. Rename Method: isQuitCommand is renamed to isQuitExitCommand.
	 */
	private boolean isQuitExitCommand(String command) {
		return ((command.equals("quit")) || (command.equals("exit")));
	}

	/**
	 * 4.2. Decompose Conditional: isHelpCommand method.
	 */
	private boolean isHelpCommand(String command) {
		return (command.equals("help"));
	}

	/**
	 * 4.2. Decompose Conditional: isShortLocalListCommand method.
	 */
	private boolean isShortLocalListCommand(String command) {
		return (command.equals(String.valueOf(llistShort1)) || command
				.equals(String.valueOf(llistShort2)));
	}
```

```
	/**
	 * 4.2. Decompose Conditional: isLongLocalListCommand method.
	 */
	private boolean isLongLocalListCommand(String command) {
		return (command.equals(String.valueOf(llistLong1)) || command
				.equals(String.valueOf(llistLong2)));
	}

	/**
	 * 4.2. Decompose Conditional: isMakeLocalDirectoryCommand method.
	 */
	private boolean isMakeLocalDirectoryCommand(String command) {
		return (command.startsWith("lmkdir"));
	}

	/**
	 * 4.2. Decompose Conditional: isRemoveLocalDirectoryCommand method.
	 */
	private boolean isRemoveLocalDirectoryCommand(String command) {
		return (command.startsWith("lrmdir"));
	}

	/**
	 * 4.2. Decompose Conditional: isPrintLocalDirectoryCommand method.
	 */
	private boolean isPrintLocalDirectoryCommand(String command) {
		return (command.equals("lpwd"));
	}

	/**
	 * 4.2. Decompose Conditional: isChangeLocalDirectoryCommand method.
	 */
	private boolean isChangeLocalDirectoryCommand(String command) {
		return (command.startsWith("lcd"));
	}

	/**
	 * 4.2. Decompose Conditional: isCloseCommand method.
	 */
	private boolean isCloseCommand(String command) {
		return (command.equals("close"));
	}

	/**
	 * 4.2. Decompose Conditional: isChangeDirectoryCommand method.
	 */
	private boolean isChangeDirectoryCommand(String command) {
		return (command.startsWith("cd"));
	}

	/**
	 * 4.2. Decompose Conditional: isShortListCommand method.
	 */
	private boolean isShortListCommand(String command) {
		return (command.equals(String.valueOf(listShort1)) || command
				.equals(String.valueOf(listShort2)));
	}

	/**
	 * 4.2. Decompose Conditional: isLongListCommand method.
	 */
	private boolean isLongListCommand(String command) {
		return (command.equals(String.valueOf(listLong1)) || command
```

```
                        .equals(String.valueOf(listLong2)));
        }

        /**
         * 4.2. Decompose Conditional: isGetCommand method.
         */
        private boolean isGetCommand(String command) {
                return (command.startsWith("get"));
        }

        /**
         * 4.2. Decompose Conditional: isMGetCommand method.
         */
        private boolean isMGetCommand(String command) {
                return (command.startsWith("mget"));
        }

        /**
         * 4.2. Decompose Conditional: isPutCommand method.
         */
        private boolean isPutCommand(String command) {
                return (command.startsWith("put"));
        }

        /**
         * 4.2. Decompose Conditional: isMPutCommand method.
         */
        private boolean isMPutCommand(String command) {
                return (command.startsWith("mput"));
        }

        /**
         * 4.2. Decompose Conditional: isTypeCommand method.
         */
        private boolean isTypeCommand(String command) {
                return (command.startsWith("type") || command.equals("bin") ||
command
                        .equals("asc"));
        }

        /**
         * 4.2. Decompose Conditional: isMakeDirectoryCommand method.
         */
        private boolean isMakeDirectoryCommand(String command) {
                return (command.startsWith("mkdir"));
        }

        /**
         * 4.2. Decompose Conditional: isRemoveDirectoryCommand method.
         */
        private boolean isRemoveDirectoryCommand(String command) {
                return (command.startsWith("rmdir"));
        }

        /**
         * 4.2. Decompose Conditional: isPrintDirectoryCommand method.
         */
        private boolean isPrintDirectoryCommand(String command) {
                return (command.equals("pwd"));
        }

        /**
         * 4.2. Decompose Conditional: isByeCommand method.
```

```
         */
        private boolean isByeCommand(String command) {
                return (command.equals("bye"));
        }
}

/*
 * Name:    Parser.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * 2.1. Extract Package: parsers package.
 */
package jftplevelrefactoring.looplevel.parsers;

/**
 * Abstract class. Defines methods to parse Strings.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.6.1. Inline Class: Inheritance to JFTPSuper is directly removed because
it
 * is not used.
 */
public abstract class Parser {
        public Parser() {
                boolean busy = false;
                System.out.println(busy);
                /*
                 * There was an error msg at compile-time that let me do such
strange
                 * things. ;-)
                 */
        }

        public abstract void parse(String s);
}

/*
 * Name:    ServerResponseParser.java (extends Parser (extends JFTPSuper))
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz Creation
 */
/**
 * 2.1. Extract Package: parsers package.
 */
package jftplevelrefactoring.looplevel.parsers;

/**
 * Parses the FTPServers responses.
 *
 * @since 0.0.2pA1
 */
public class ServerResponseParser extends Parser {
        /**
```

```
         * Parses a given String for known FTPServer-replies.
         *
         * @param s
         *            String to parse
         *@since 0.0.2pA1
         */
        public void parse(String s) {
               /*
                *  if  (s.equals(xxxxxx))  {  Insert  known  FTPServer  replies
here. }
                */
        }
}

/*
 * Name:    Progressbar.java  (extens JFTPSuper.java)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name         changes
 * 0.1.99   11.06.2001 Tobias Kranz Creation
 * 0.1.99t5 23.07.2001 Tobias Kranz Added Comments
 */
/**
 * 2.1. Extract Package: ui package.
 */
package jftplevelrefactoring.looplevel.ui;

/**
 * Shows a textual progressbar
 *
 * @since 0.1.99
 *@version 1
 */
public class Progressbar {
        private long max;
        private long pos = 1;

        /**
         * Constructor. Here you must set the maximum value the progressbar
may
         * have.
         *
         * @param max
         *            The maximum value the progressbar may have.
         *@since 0.1.99
         *@version 1
         */
        public Progressbar(int max) {
               this.max = max;
               this.show();
        }

        /**
         * Sets the progressbar to the given value.
         *
         * @param pos
         *            The current position of the progressbar
         *@since 0.1.99
         *@version 1
         */
        public void set(int pos) {
               if (pos <= max) {
```

```
                    this.pos = pos;
             }
             show();
      }

      /**
       * Shows the progressbar in it's current status
       *
       * @since 0.1.99
       *@version 1
       */
      private void show() {
             int i, stat = 0;
             String graphStat = "";
             double stat1 = ((double) 100 / max) * pos;

             String statstr = Double.toString(stat1);
             stat          =          Integer.parseInt(statstr.substring(0,
statstr.indexOf('.')));

             for (i = 0; i < (int) 20 * (((float) stat) / 100); i++)
                    graphStat += "="; // setting '='s for reached %

             for (; i < 20; i++)
                    graphStat += " "; // filling the rest with ' 's

             if ((int) ((float) stat) / 100 >= 1)
                    System.out.print("Progress: [" + graphStat + "]: " +
stat
                                  + "% complete (" + (int) pos + " bytes
transfered)\n");
             // else if ((int)((float)stat)/10 >= 1)
             //  System.out.print("Progress:  ["+graphStat+"]:    "  +stat+"%
complete ("+
             // (int)pos/1024+" kb)\r");
             else
                    System.out.print("Progress: [" + graphStat + "]:    " +
stat
                                  + "% complete (" + (int) pos / 1024 + "
kb)\r");
      }
}

/*
 * Name:    JFTP.java (extends JFTPSuper)
 * Author:  JavaFTP-Group
 * License: GPL
 *
 * version  date       name                      changes
 * 0.0.2pA1 10.04.2001 Tobias Kranz              Creation
 * 0.0.2pA2 02.05.2001 Tobias Kranz              Added 'nextParamIsNumeric()'
 *                                               & 'nextParamExists()'
 * 0.0.2pA2 04.05.2001 Sebastian Schipper        Two bugs fixed
 */
package jftplevelrefactoring.looplevel;

import java.io.IOException;
import java.util.BitSet;

import jftplevelrefactoring.looplevel.core.FTPAuthentication;
import jftplevelrefactoring.looplevel.io.StdErr;
import jftplevelrefactoring.looplevel.io.StdIn;
import jftplevelrefactoring.looplevel.io.StdOut;
```

```
import jftplevelrefactoring.looplevel.parsers.CommandLineParser;
import jftplevelrefactoring.looplevel.parsers.Parser;

/**
 * Main class.
 *
 * @since 0.0.2pA1
 */
/**
 * 1.1.3.2. A local variable of JFTPSuper to cover jftpSuper. 1.6.1. Inline
 * Class: Inheritance to JFTPSuper is directly removed. 1.6.2. Inheritance
 * relationship usage is replaced using a local variable of JFTPSuper.
 */
public class JFTP {

	private final char[] guiShort = { '-', 'g' };
	private final char[] cliShort = { '-', 'c' };
	private final char[] helpShort = { '-', 'h' };
	private final char[] portShort = { '-', 'P' };
	private final char[] passiveShort = { '-', 'p' };
	private final char[] verboseShort = { '-', 'v' };
	private final char[] anonymousShort = { '-', 'a' };
	private final char[] aPasswdShort = { '-', 'A', 'p' };
	private final char[] guiLong = { '-', '-', 'g', 'u', 'i' };
	private final char[] cliLong = { '-', '-', 'c', 'l', 'i' };
	private final char[] helpLong = { '-', '-', 'h', 'e', 'l', 'p' };
	private final char[] portLong = { '-', '-', 'P', 'o', 'r', 't' };
	private final char[] verboseLong = { '-', '-', 'v', 'e', 'r', 'b',
'o',
			's', 'e' };
	private final char[] passiveLong = { '-', '-', 'p', 'a', 's', 's',
'i',
			'v', 'e' };
	private final char[] anonymousLong = { '-', '-', 'a', 'n', 'o', 'n',
'y',
			'm', 'o', 'u', 's' };
	private final char[] aPasswdLong = { '-', '-', 'A', 'n', 'o', 'n',
'y',
			'm', 'o', 'u', 's', 'P', 'a', 's', 's', 'w', 'd' };

	/**
	 * 3.4. Move Field: options local variable is moved from JFTPSuper to
	 * replace inheritance.
	 */
	private BitSet options = new BitSet(4);

	/**
	 * 1.2.3.3. A local variable of FTPAuthentication.
	 */
	private FTPAuthentication ftpAuthentication;

	private StdIn stdin;
	private StdOut stdout;
	private StdErr stderr;

	/**
	 * Sets initial values.
	 *
	 * @since 0.0.2pA1
	 */
	public JFTP() {
		this.ftpAuthentication = new FTPAuthentication();
		this.stdin = new StdIn();
```

```
                this.stdout = new StdOut();
                this.stderr = new StdErr();
                /*
                 * BitSet 'options'
                 *
                 * #: if true: default: 0 => cli true, 1 => passive true, 2 =>
anonymous
                 * false, 3 => autoconnect false.
                 */
                this.options.set(0);
                this.options.set(1);
                this.options.clear(2);
                this.options.clear(3);
        }

        /**
         * Is Cli-mode running ?
         *
         * @return true if Cli-mode is running; false if not
         *@since 0.0.2pA1
         */
        protected boolean isCli() {
                return ((boolean) options.get(0));
        }

        /**
         * Sets Cli-mode to true.
         *
         * @since 0.0.2pA1
         */
        protected void setCli() {
                options.set(0);
        }

        /**
         * Sets Cli-mode to false.
         *
         * @since 0.0.2pA1
         */
        protected void unsetCli() {
                options.clear(0);
        }

        /**
         * Is Gui-mode running ?
         *
         * @return true if isCli() returns false and other way round.
         *@since 0.0.2pA1
         */
        protected boolean isGui() {
                return (((boolean) options.get(0) ? (false) : (true)));
        }

        /**
         * Same than unsetCli.
         *
         * @since 0.0.2pA1
         */
        protected void setGui() {
                unsetCli();
        }

        /**
```

```
         * Is passive connection mode prefered ?
         *
         * @return true if passive mode is prefered; false if not
         *@since 0.0.2pA1
         */
        protected boolean isPassive() {
               return ((boolean) options.get(1));
        }

        /**
         * Sets passive mode to true.
         *
         * @since 0.0.2pA1
         */
        protected void setPassive() {
               options.set(1);
        }

        /**
         * Sets passive mode to false.
         *
         * @since 0.0.2pA1
         */
        protected void unsetPassive() {
               options.clear(1);
        }

        /**
         * Is anonymous logon prefered ?
         *
         * @return true if anonymous logon is prefered; otherwise false
         *@since 0.0.2pA1
         */
        protected boolean isAnonymous() {
               return ((boolean) options.get(2));
        }

        /**
         * Sets anonymous logon to true.
         *
         * @since 0.0.2pA1
         */
        protected void setAnonymous() {
               options.set(2);
        }

        /**
         * Sets anonymous logon to false.
         *
         * @since 0.0.2pA1
         */
        protected void unsetAnonymous() {
               options.clear(2);
        }

        /**
         * Returns Autoconnect
         *
         * @return true if Autoconnect is true; otherwise false
         *@since 0.0.2pA2
         */
        protected boolean isAutoconnect() {
               return ((boolean) options.get(3));
```

```
	}

	/**
	 * Sets Autoconnect to true.
	 *
	 * @since 0.0.2pA2
	 */
	protected void setAutoconnect() {
		options.set(3);
	}

	/**
	 * Sets Autoconnect to false.
	 *
	 * @since 0.0.2pA2
	 */
	protected void unsetAutoconnect() {
		options.clear(3);
	}

	/**
	 * Starts the prog.
	 *
	 * @param CommandLineArguments
	 *                .
	 *@since 0.0.2pA1
	 */
	public static void main(String[] args) {
		JFTP j = new JFTP();
		j.parseArgs(args); // scanning cmdLine

		if (j.isCli())
			j.cli();
		else
			j.gui();
	}

	/**
	 * Starts the CommandLineInterface (cli) version.
	 *
	 * @since 0.0.2pA1
	 */
	private void cli() {
		Parser clp = new CommandLineParser();

		if (ftpAuthentication.getDebugLevel() >= 1)
			printOutputLine("DebugLevel:             "             +
ftpAuthentication.getDebugLevel());
		printOutputLine(ftpAuthentication.getJFTPInfo());
		printOutputLine(ftpAuthentication.getCopyright());
		printOutputLine(ftpAuthentication.getLicenseInfo());

		if (isAutoconnect()) {
			clp.parse("open " + ftpAuthentication.getServerIP());
		}

		while (true) {
			printOutput("jftp> ");

			try {
				clp.parse(readInputLine());
			} catch (IOException e) {
				printErrorLine("Couldn't read from StdIn!");
```

```
                }
            }
        }

        /**
         * Starts the GUI version.
         *
         * @since 0.0.2pA1
         */
        private void gui() {
            // GUI gui1 = new GUI();
        }

        /**
         * Parses the commandLine arguments.
         *
         * @param a
         *            String array to parse
         *@since 0.0.2pA1
         *@version 1
         */
        private void parseArgs(String[] a) {
            int i;

            try {
                for (i = 0; i < a.length; i++) {
                    if (isHelpCommand(a[i])) {
                        this.showHelp();
                    } // Help
                    else if (isPassiveCommand(a[i])) {
                        setPassive();
                    } // Passive mode
                    else if (isAnonymousCommand(a[i])) {
                        setAnonymous();
                    } // Anonymous mode
                    else if (isAnonymousPasswordCommand(a[i])) { //
Anonymous
                        // Passwd
                        if (isAnonymous()) {
                            if (this.nextParamExists(a, i))

        ftpAuthentication.setAnonymousPasswdTo(a[++i]);
                            else
                                this.showHelp();
                        } else {
                            printErrorLine("You have set the
anonymous password but not anonymous login!?!?");
                            printErrorLine("Exiting with error
(1).");
                            System.exit(1);
                        }

                        if (ftpAuthentication.getDebugLevel() >= 2)
                            printOutputLine("Anonymous Password:
"
                                    +
ftpAuthentication.getAnonymousPasswd());
                    } else if (isGUICommand(a[i])) {
                        setGui();
                    } // GUI
                    else if (isCLICommand(a[i])) {
                        setCli();
                    } // CLI
```

```
                        else if (isVerbosityCommand(a[i])) { // Verbosity
                                if      ((i      <      a.length)      &&
(this.nextParamIsNumeric(a, i))) {
                                        short     tmpDebug     =     (short)
Integer.parseInt(a[++i]);
                                        if ((tmpDebug <= 2) && (tmpDebug >=
0))

      ftpAuthentication.setDebugLevel(tmpDebug);
                                } else {
                                        printErrorLine("Missing   or   wrong
parameter.");
                                        this.showHelp();
                                }
                        } // Port
                        else if (isPortCommand(a[i])) {
                                if      ((i      <      a.length)      &&
(this.nextParamIsNumeric(a, i))) {
                                        int             tmpPort             =
Integer.parseInt(a[++i]);
                                        if ((tmpPort >= 1) && (tmpPort <=
65535)) // Ok !
                                        {

      ftpAuthentication.setServerPortTo(tmpPort);
                                                printOutputLine("Don't use std
port. Using "
                                                                +
ftpAuthentication.getServerPort()
                                                                + " instead.");
                                        } else {
                                                printOutputLine("!WARNING!
Specified port not in proper range.");
                                                printOutputLine("!WARNING!
Falling back to default port (21).");
                                        }
                                } else {
                                        printErrorLine("Missing   or   wrong
parameters");
                                        this.showHelp();
                                }
                        } else {
                                if (i == (a.length - 1)) // Last parameter
                                {
                                        if
(ftpAuthentication.setServerTo(a[i]))
                                                setAutoconnect();
                                } else {
                                        printOutputLine("Unknown  Option:  '"
+ a[i] + "'");
                                        this.showHelp();
                                }
                        }
                }

            } catch (Exception e) {
                  printOutput("Exception: ");
                  e.printStackTrace();
            }

            // if (ftpConnection.isPassive())
            // printOutputLine("PASSIVE mode prefered.");
            if (isAnonymous())
```

```
                    printOutputLine("We'll logon as \"anonymous\".");
              // if (ftpConnection.isCli()) printOutputLine("starting CLI.");
              if (isGui())
                    printOutputLine("starting GUI.");
       }

       /**
        * 4.8. Remove Assignments to Parameters: assignments of parameter of
        * nextParamIsNumeric method are removed by using temporary variable.
        */
       /**
        * Checks if the next parameter is a numeric value.
        *
        * @param a
        *            String array of parameters to check
        *@param pos
        *            current position in array a
        *@return true if the next parameter exists and is numeric;otherwise
false
        *@since 0.0.2pA2
        */
       private boolean nextParamIsNumeric(String[] a, int pos) {
              boolean rc = false;
              int tmp;

              /**
               * 4.8.1. Temporary variable is set value of parameter. 4.8.2.
All
               * assignments to parameter are set to temporary variable.
               */
              int temp = pos;
              try {
                    tmp = Integer.parseInt(a[(++temp)]);
                    writeLineToConsole(tmp + "");
                    rc = true;
              } catch (IndexOutOfBoundsException iobe) {
                    if (ftpAuthentication.getDebugLevel() >= 2)
                          printErrorLine("No more parameters.");
              } catch (NumberFormatException nfe) {
                    if (ftpAuthentication.getDebugLevel() >= 2)
                          printErrorLine("No more parameters.");
              }

              return rc;
       }

       /**
        * Checks if the next parameter exists.
        *
        * @param a
        *            String array to check
        *@param pos
        *            current position in array a
        *@return true if the next parameter exists; else false
        *@since 0.0.2pA2
        */
       private boolean nextParamExists(String[] a, int pos) {
              boolean rc = true;

              int temp = pos;
              try {
                    String tmp = a[++temp];
                    writeLineToConsole(tmp);
```

```
            } catch (IndexOutOfBoundsException iobe) {
                  rc = false;
            }

            // maybe "(pos < a.size) ? (return true) : (return false)

            return rc;
      }

      /**
       * Shows the help screen.
       *
       * @since 0.0.1
       */
      private void showHelp() {
            /*
             * Verbosity level > 1 should only be used for development.
Stable
             * versions should only contain an option for setting verbosity
to true
             * or false.
             */
            printOutputLine(ftpAuthentication.getJFTPInfo());
            printOutputLine("usage: JFTP [-h[c|g]pa] [Server]");
            printOutputLine("       JFTP [--help][--verbose X][--passive][-
-anonymous[--anonymousPasswd X]]");
            printOutputLine("            [--port X][Server]\n");
            printOutputLine("       -h,    --help                Shows this
help");
            printOutputLine("       -c,    --cli                 Starts the
CLI-version (Default).");
            printOutputLine("       -g,    --gui                 Starts the
GUI-version.");
            printOutputLine("         -p,     --passive               Force
passive mode ftp.");
            printOutputLine("                                      (Default is
active mode ftp(NOT now))");
            printOutputLine("       -a,    --anonymous          Causes jftp
to bypass normal login procedure");
            printOutputLine("                                        and use
anonymous login instead.");
            printOutputLine("       -Ap X, --AnonymousPasswd X  Causes jftp
to use X as anonymous password.");
            printOutputLine("        -v X,   --verbose X           Sets the
vorbosity level where X = 0 - 2.");
            printOutputLine("        -P X,   --Port X              Sets the
port number to X.");
            System.exit(0); // <- !EXIT!
      }

      /**
       * 4.1. Extract Method: readInputLine, printOutput, printOutputLine,
       * printErrorLine and writeLineToConsole methods.
       */
      private String readInputLine() throws IOException {
            return stdin.readLine();
      }

      private void printOutput(String message) {
            stdout.print(message);
      }

      private void printOutputLine(String message) {
```

```
                stdout.println(message);
        }

        private void printErrorLine(String message) {
                stderr.println(message);
        }

        protected void writeLineToConsole(String message) {
                System.out.println(message);
        }

        /**
         * 4.2. Decompose Conditional: isHelpCommand method.
         */
        private boolean isHelpCommand(String command) {
                return (command.equals(String.valueOf(helpShort)) || command
                                .equals(String.valueOf(helpLong)));
        }

        /**
         * 4.2. Decompose Conditional: isPassiveCommand method.
         */
        private boolean isPassiveCommand(String command) {
                return (command.equals(String.valueOf(passiveShort)) || command
                                .equals(String.valueOf(passiveLong)));
        }

        /**
         * 4.2. Decompose Conditional: isAnonymousCommand method.
         */
        private boolean isAnonymousCommand(String command) {
                return    (command.equals(String.valueOf(anonymousShort))    ||
command
                                .equals(String.valueOf(anonymousLong)));
        }

        /**
         * 4.2. Decompose Conditional: isAnonymousPasswordCommand method.
         */
        private boolean isAnonymousPasswordCommand(String command) {
                return (command.equals(String.valueOf(aPasswdShort)) || command
                                .equals(String.valueOf(aPasswdLong)));
        }

        /**
         * 4.2. Decompose Conditional: isGUICommand method.
         */
        private boolean isGUICommand(String command) {
                return (command.equals(String.valueOf(guiShort)) || command
                                .equals(String.valueOf(guiLong)));
        }

        /**
         * 4.2. Decompose Conditional: isCLICommand method.
         */
        private boolean isCLICommand(String command) {
                return (command.equals(String.valueOf(cliShort)) || command
                                .equals(String.valueOf(cliLong)));
        }

        /**
         * 4.2. Decompose Conditional: isVerbosityCommand method.
         */
```

```
        private boolean isVerbosityCommand(String command) {
               return (command.equals(String.valueOf(verboseShort)) || command
                              .equals(String.valueOf(verboseLong)));
        }

        /**
         * 4.2. Decompose Conditional: isPortCommand method.
         */
        private boolean isPortCommand(String command) {
               return (command.equals(String.valueOf(portShort)) || command
                              .equals(String.valueOf(portLong)));
        }
}
```

Source Code B-7: JFTP Refactored Source Code Using LRtF

REFERENCES

[1] D. Leffingwell and D. Widrig, *Managing Software Requirements: a Unified Approach*, First ed.: Addison-Wesley Longman Publishing Co., Inc., 2000.

[2] M. Fayad and D. Schmidt, "Object-Oriented Application Frameworks," *Communications of the ACM,* vol. 40, October, 1997 1997.

[3] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*, First ed.: Addison-Wesley Longman Publishing Co., Inc., 1999.

[4] M. Fayad, D. Schmidt, and R. Johnson, *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, First ed.: John Wiley and Sons, Inc., 1999.

[5] M. Fayad, D. Schmidt, and R. Johnson, *Implementing Application Frameworks: Object-Oriented Frameworks at Work*, First ed.: John Wiley and Sons, Inc., 1999.

[6] R. E. Johnson, "How Frameworks Compare To Other Object-Oriented Reuse Techniques: Frameworks = (Components + Patterns)," *Communications of the ACM,* vol. 40, October, 1997 1997.

[7] W. Wake, *Refactoring Workbook*, First ed.: Addison-Wesley Longman Publishing Co., Inc., 2003.

[8] H. Liu, G. Li, Z. Ma, and W. Shao, "Conflict-Aware Schedule of Software Refactorings," *IET Software,* vol. 2, pp. 446-460, October, 2008 2008.

[9] S. Bryton and F. Abreu, "Modularity-Oriented Refactoring," in *The 12th European Conference on Software Maintenance and Reengineering (CSMR'08)* Athens, Greece: IEEE, 2008, pp. 294-297.

[10] T. Mens and T. Tourwe, "A Survey of Software Refactoring," *IEEE Transactions on Software Engineering,* vol. 30, February, 2004 2004.

[11] R. Tiarks, "Quality-Driven Refactoring," in *Informatic Seminar Transformation (IST'05)* University of Bremen, Bremen, Germany, 2005.

[12] L. Tahvildari, "Quality-Driven Object-Oriented Re-Engineering Framework," in *The 20th IEEE International Conference on Software Maintenance (ICSM'04)* Chicago, IL, USA: IEEE Computer Society, 2004, pp. 479-483.

[13] L. Tahvildari, K. Kontogiannis, and J. Mylopoulos, "Quality-Driven Software Re-Engineering," *Journal of Systems and Software,* vol. 66, pp. 225-239, June, 2003 2003.

[14] B. Geppert, A. Mockus, and F. Rossler, "Refactoring for Changeability: A Way to Go?," in *The 11th IEEE International Software Metrics Symposium (METRICS'05)* Como, Italy: IEEE Computer Society, 2005.

[15] R. Kolb, D. Muthig, T. Patzke, and K. Yamauchi, "A Case Study in Refactoring a Legacy Component for Reuse in a Product Line," in *The 21st IEEE International Conference on Software Maintenance (ICSM'05)* Budapest, Hungary: IEEE Computer Society, 2005, pp. 369-378.

[16] J. Lee, N. Lee, and S. Rhew, "Object-Oriented Refactorng Process Design for the Software Reuse," in *IEEE International Symposium on Industrial Electronics (ISIE'01)*, 2001.

[17] Z. Xing and S. Eleni, "Refactoring Practice: How it is and How it Should be Supported - An Eclipse Case Study," in *The 22nd IEEE International Conference on Software Maintenance (ICSM'06)* Philadelphia, Pennsylvania, USA, 2006, pp. 458-468.

[18] M. Goldstein, Y. A. Feldman, and S. Tyszberowicz, "Refactoring with Contracts," in *IEEE Agile Conference (AGILE'06)* Minneapolis, Minnesota, USA: IEEE Computer Society, 2006, pp. 53-64.

[19] N. Ubayashi, J. Piao, S. Shinotsuka, and T. Tamai, "Contract-Based Verification for Aspect-Oriented Refactoring," in *The 2008 International Conference on Software Testing, Verification, and Validation (ICST'08)* Lillehammer, Norway: IEEE Computer Society, 2008, pp. 180-189.

[20] S. Shrivastava and V. Shrivastava, "Impact of Metrics Based Refactoring on the Software Quality: a Case Study," in *The 2008 IEEE Region 10 Conference (TENCON'08)* Hyderabad, India, 2008, pp. 1-6.

[21] N. Tsantalis and A. Chatzigeorgiou, "Identification of Extract Method Refactoring Opportunities," in *The 13th European Conference on Software Maintenance and Reengineering (CSMR'09)* Kaiserslautern, Germany, 2009, pp. 119-128.

[22] I. Ivkovic and K. Kontogiannis, "A Framework for Software Architecture Refactoring using Model Transformations and Semantic Annotations," in *The Conference on Software Maintenance and Reengineering (CSMR'06)* Bari, Italy: IEEE Computer Society, 2006, pp. 135-144.

[23] L. Grunske and R. Neumann, "Process Components for Quality Evaluation and Quality Improvement," in *The 2nd Workshop on Method Engineering for Object-Oriented and Component-Based Development (OOPSLA'04), The International Conference on Object Oriented Programming, Systems, Languages and Applications*, 2004.

[24] J. Rech and E. Ras, "Experience-Based Refactoring for Goal-Oriented Software Quality Improvement," in *First International Workshop on Software Quality (SOQUA'04)*, 2004.

[25] D. Dig and R. Johnson, "The Role of Refactorings in API Evolution," in *The 21st IEEE International Conference on Software Maintenance (ICSM'05)* Budapest, Hungary: IEEE Computer Society, 2005, pp. 389-398.

[26] G. Butler and L. Xu, "Cascaded Refactoring for Framework Evolution," *Special Interest Group on Software Engineering (SIGSOFT) Software Engeering Notes,* vol. 26, pp. 51-57, May, 2001 2001.

[27] E. Murphy-Hill and A. Black, "Refactoring Tools: Fitness for Purpose," *IEEE Software,* vol. 25, pp. 38-44, September-October, 2008 2008.

[28] R. Marticorena, C. Lopez, Y. Crespo, and F. J. Perez, "Reuse Based Refactoring Tools," in *The 1st Workshop on Refactoring Tools (WRT'07), The 21st European Conference on Object-Oriented Programming (ECOOP'07)* Berlin, Germany, 2007.

[29] E. Murphy-Hill and A. Black, "Breaking the Barriers to Successful Refactoring: Observations and Tools for Extract Method," in *ACM/IEEE 30th International Conference on Software Engineering (ICSE'08)* Leipzig, Germany, 2008, pp. 421-430.

[30] N. C. Mendonca, P. H. Maia, L. A. Fonseca, and R. M. Andrade, "RefaX: A Refactoring Framework Based on XML," in *The 20th IEEE International*

Conference on Software Maintenance (ICSM'04) Chicago, IL, USA: IEEE Computer Society, 2004, pp. 147-156.

[31] K. Maruyama and S. Yamamoto, "Design and Implementation of an Extensible and Modifiable Refactoring Tool," in *The 13th International Workshop on Program Comprehension (IWPC'05)* St. Louis, MO, USA: IEEE Computer Society, 2005, pp. 195-204.

[32] S. Lai and C. Yang, "A Software Metric Combination Model for Software Reuse," in *Fifth Asia-Pacific Software Engineering Conference (APSEC'98)* Taipei, Taiwan, 1998.

[33] F. Dandashi, "A Method for Assessing the Reusability of Object-Oriented Code Using a Validated Set of Automated Measurements," in *The 2002 ACM Symposium on Applied Computing (SAC'02)* Madrid, Spain, 2002.

[34] J. Perry, "Perspective on Software Reuse," Software Engineering Institute (SEI), Technical Report CMU/SEI-88-SR-022 September, 1988 1988.

[35] J. Estublier and G. Vega, "Reuse and Variability in Large Software Applications," *Special Interest Group on Software Engineering (SIGSOFT) Software Engeering Notes,* vol. 30, pp. 316-325, September, 2005 2005.

[36] T. Lopes, I. Neag, and J. Ralph, "The Role of Extensibility in Software Standards for Automatic Test Systems," in *IEEE Autotestcon (AUTOTESTCON'05)*, 2005, pp. 367-373.

[37] F. Zhuo, B. Lowther, P. Oman, and J. Hagemeister, "Constructing and Testing Software Maintainability Assessment Models," in *First International Software Metrics Symposium (METRICS'93)* Baltimore, Maryland, USA: IEEE Computer Society, 1993, pp. 61-70.

[38] R. Land, "Measurements of Software Maintainability," in *Graduate Student Conference* Uppsala University, Uppsala, Sweden, 2002.

[39] N. Bevan, "Measuring Usability as Quality of Use," *Journal of Software Quality,* vol. 4, pp. 115-130, March, 1995 1995.

[40] N. Bevan and M. Macleod, "Usability Measurement in Context," *Behaviour and Information Technology (BIT),* vol. 13, pp. 132-145, 1994.

[41] M. Bertoa and A. Vallecillo, "Usability Metrics for Software Components," in *Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'04), The 8th European Conference on Object-Oriented Programming (ECOOP'04)* Oslo, Norway, 2004.

[42] A. Eden and T. Mens, "Measuring Software Flexibility," *IEE Proceedings Software,* vol. 153, pp. 113-126, June, 2006 2006.

[43] S. Kan, *Metrics and Models in Software Quality Engineering*, Second ed.: Addison-Wesley Longman Publishing Co., Inc., 2002.

[44] N. Fenton and S. Pfleeger, *Software Metrics: A Rigorous Approach*, Second ed. London: Int'l Thomson Computer Press, 1996.

[45] J. A. McCall, P. K. Richards, and G. F. Walters, "Factors in Software Quality," US Rome Air Development Center 1977.

[46] B. W. Boehm, J. R. Brown, and J. R. Kaspar, *Characteristics of Software Quality (TRW Series of Software Technology)*, First ed. Amsterdam, North Holland: Elsevier Science Ltd, 1978.

[47] *Standard for a Software Quality Metrics Methodology*: IEEE Std 106 1-1 992, 1993.

[48] L. Briand and W. J., "Empirical Studies of Quality Models in Object-Oriented Systems," *Advances in Computers, Academic Press,* 2002.
[49] M. Zenger, "KERIS: evolving software with extensible modules: Research Articles," *Journal of Software Maintenance and Evolution: Research and Practice,* vol. 17, pp. 333-362, September, 2005 2005.
[50] M. Fowler, "Refactoring Catalog: Refactorings in Alphabetical Order," 2008.
[51] K. Rubin, "Reuse in Software Engineering: an Object-Oriented Perspective," in *IEEE COMPCON*, 1990, pp. 340-346.
[52] L. Bass, B. John, and J. Kates, "Achieving Usability through Software Architecture," Software Engineering Institute (SEI), Technical Report CMU/SEI-2001-TR-005 March, 2001 2001.
[53] SJEA, "Simple JAVA Encryption Algorithm (SJEA)," SourceForge, 2009.
[54] JFTP, "JAVA File Transfer Protocol Client (JFTP)," SourceForge, 2001.
[55] SourceForge, "Source Forge: Find and Build Open Source Software," 2009.
[56] MetaMata, "MetaMata Metrics Tool v2.0," MetaMata, Inc., 2000.

VITA

Name: Faisal Mohammed Abobakr Banaeamah

Date of Birth: 20/11/1983

Nationality: Yemeni

E-Mail Address: fmb@kfupm.edu.sa, mr.fmab@gmail.com

Faisal Mohammed Banaeamah received his B. S. in Software Engineering, with second honor, from King Fahd University of Petroleum and Minerals (KFUPM) in June 2006. In his summer training project, he worked in development of Online Press Request System (OPRS) for KFUPM Press. He also continued with KFUPM Press after his graduation for one year to complete the project.

Since then, he has worked as a research assistant for the Information and Computer Science Department at KFUPM while pursuing his M. S. in Computer Science. He took advanced computer science courses such as Software Design, Database Design and Implementation, Artificial Intelligence, and Computer Algorithms.

After he finished the graduate courses, he converted his academic status from full time to part time and joined to the Information and Technology Center at KFUPM and worked as E-Business HR Consultant. His research interests include application frameworks, software refactoring, software design, and RDBMS to XML mapping.